# Starbucks_Capstone_notebook

July 12, 2021

# 1 Starbucks Capstone Challenge

### 1.0.1 Introduction

This data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. 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). Some users might not receive any offer during certain weeks.

Not all users receive the same offer, and that is the challenge to solve with this data set.

Your task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products.

Every offer has a validity period before the offer expires. As an example, a BOGO offer might be valid for only 5 days. You'll see in the data set that informational offers have a validity period even though these ads are merely providing information about a product; for example, if an informational offer has 7 days of validity, you can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement.

You'll be given transactional data showing user purchases made on the app including the timestamp of purchase and the amount of money spent on a purchase. This transactional data also has a record for each offer that a user receives as well as a record for when a user actually views the offer. There are also records for when a user completes an offer.

Keep in mind as well that someone using the app might make a purchase through the app without having received an offer or seen an offer.

### 1.0.2 Example

To give an example, a user could receive a discount offer buy 10 dollars get 2 off on Monday. The offer is valid for 10 days from receipt. If the customer accumulates at least 10 dollars in purchases during the validity period, the customer completes the offer.

However, there are a few things to watch out for in this data set. Customers do not opt into the offers that they receive; in other words, a user can receive an offer, never actually view the offer, and still complete the offer. For example, a user might receive the "buy 10 dollars get 2 dollars off offer", but the user never opens the offer during the 10 day validity period. The customer spends 15 dollars during those ten days. There will be an offer completion record in the data set; however, the customer was not influenced by the offer because the customer never viewed the offer.

### 1.0.3 Cleaning

This makes data cleaning especially important and tricky.

You'll also want to take into account that some demographic groups will make purchases even if they don't receive an offer. From a business perspective, if a customer is going to make a 10 dollar purchase without an offer anyway, you wouldn't want to send a buy 10 dollars get 2 dollars off offer. You'll want to try to assess what a certain demographic group will buy when not receiving any offers.

### 1.0.4 Final Advice

Because this is a capstone project, you are free to analyze the data any way you see fit. For example, you could build a machine learning model that predicts how much someone will spend based on demographics and offer type. Or you could build a model that predicts whether or not someone will respond to an offer. Or, you don't need to build a machine learning model at all. You could develop a set of heuristics that determine what offer you should send to each customer (i.e., 75 percent of women customers who were 35 years old responded to offer A vs 40 percent from the same demographic to offer B, so send offer A).

# 2 Data Sets

The data is contained in three files:

- portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
- profile.json - demographic data for each customer
- transcript.json - records for transactions, offers received, offers viewed, and offers completed

Here is the schema and explanation of each variable in the files:

**portfolio.json** * id (string) - offer id * offer_type (string) - type of offer ie BOGO, discount, informational * difficulty (int) - minimum required spend to complete an offer * reward (int) - reward given for completing an offer * duration (int) - time for offer to be open, in days * channels (list of strings)

**profile.json** * age (int) - age of the customer * became_member_on (int) - date when customer created an app account * gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F) * id (str) - customer id * income (float) - customer's income

**transcript.json** * event (str) - record description (ie transaction, offer received, offer viewed, etc.) * person (str) - customer id * time (int) - time in hours since start of test. The data begins at time t=0 * value - (dict of strings) - either an offer id or transaction amount depending on the record

**Note:** If you are using the workspace, you will need to go to the terminal and run the command `conda update pandas` before reading in the files. This is because the version of pandas in the workspace cannot read in the transcript.json file correctly, but the newest version of pandas can. You can access the termnal from the orange icon in the top left of this notebook.

You can see how to access the terminal and how the install works using the two images below. First you need to access the terminal:

Then you will want to run the above command:

Finally, when you enter back into the notebook (use the jupyter icon again), you should be able to run the below cell without any errors.

In [3]:
```python
#regular libraries
import pandas as pd
import numpy as np
import math
import json

#plot libraries
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import seaborn as sns
% matplotlib inline

#machine learning library
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
#warnings ignore
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# read in the json files
portfolio = pd.read_json('data/portfolio.json', orient='records', lines=True)
profile = pd.read_json('data/profile.json', orient='records', lines=True)
transcript = pd.read_json('data/transcript.json', orient='records', lines=True)
pd.set_option('display.max_columns', None)
```

In [4]:
```python
print('\n'.join(f'{m.__name__}=={m.__version__}' for m in globals().values() if getattr(
```

```
pandas==0.23.3
numpy==1.12.1
json==2.0.9
seaborn==0.8.1
```

## 3 Data Understanding

In [5]:
```python
portfolio.head()
```

Out[5]:
```
                         channels  difficulty  duration  \
0       [email, mobile, social]          10         7
1  [web, email, mobile, social]          10         5
2          [web, email, mobile]           0         4
3          [web, email, mobile]           5         7
```

```
4             [web, email]          20         10

                                 id  offer_type  reward
0  ae264e3637204a6fb9bb56bc8210ddfd           bogo      10
1  4d5c57ea9a6940dd891ad53e9dbe8da0           bogo      10
2  3f207df678b143eea3cee63160fa8bed  informational       0
3  9b98b8c7a33c4b65b9aebfe6a799e6d9           bogo       5
4  0b1e1539f2cc45b7b9fa7c272da2e1d7       discount       5
```

In [6]:
```python
portfolio.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
channels      10 non-null object
difficulty    10 non-null int64
duration      10 non-null int64
id            10 non-null object
offer_type    10 non-null object
reward        10 non-null int64
dtypes: int64(3), object(3)
memory usage: 560.0+ bytes
```

In [7]:
```python
profile.head()
```

Out[7]:

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

In [8]:
```python
profile.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 5 columns):
age                 17000 non-null int64
became_member_on    17000 non-null int64
gender              14825 non-null object
id                  17000 non-null object
income              14825 non-null float64
dtypes: float64(1), int64(2), object(2)
memory usage: 664.1+ KB
```

In [9]:
```python
profile.describe(include='all')
```

```
Out[9]:                  age  became_member_on gender  \
        count   17000.000000      1.700000e+04  14825
        unique           NaN               NaN      3
        top              NaN               NaN      M
        freq             NaN               NaN   8484
        mean       62.531412      2.016703e+07    NaN
        std        26.738580      1.167750e+04    NaN
        min        18.000000      2.013073e+07    NaN
        25%        45.000000      2.016053e+07    NaN
        50%        58.000000      2.017080e+07    NaN
        75%        73.000000      2.017123e+07    NaN
        max       118.000000      2.018073e+07    NaN

                                              id         income
        count                              17000   14825.000000
        unique                             17000            NaN
        top     0acca8aae113433999f7de6a5c32497c            NaN
        freq                                   1            NaN
        mean                                 NaN   65404.991568
        std                                  NaN   21598.299410
        min                                  NaN   30000.000000
        25%                                  NaN   49000.000000
        50%                                  NaN   64000.000000
        75%                                  NaN   80000.000000
        max                                  NaN  120000.000000
```

```
In [10]: transcript.head()
```

```
Out[10]:            event                            person  time  \
         0  offer received  78afa995795e4d85b5d9ceeca43f5fef     0
         1  offer received  a03223e636434f42ac4c3df47e8bac43     0
         2  offer received  e2127556f4f64592b11af22de27a7932     0
         3  offer received  8ec6ce2a7e7949b1bf142def7d0e0586     0
         4  offer received  68617ca6246f4fbc85e91a2a49552598     0

                                                      value
         0  {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
         1  {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
         2  {'offer id': '2906b810c7d4411798c6938adc9daaa5'}
         3  {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
         4  {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}
```

```
In [11]: transcript.info()
```

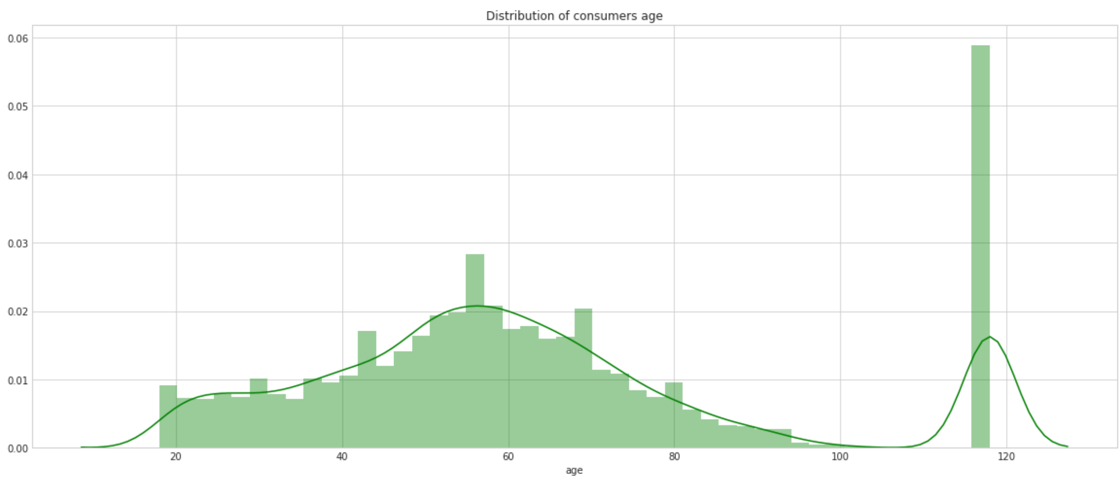
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306534 entries, 0 to 306533
Data columns (total 4 columns):
event     306534 non-null object
person    306534 non-null object
```

```
time       306534 non-null int64
value      306534 non-null object
dtypes: int64(1), object(3)
memory usage: 9.4+ MB
```

# 4 Data Exploration

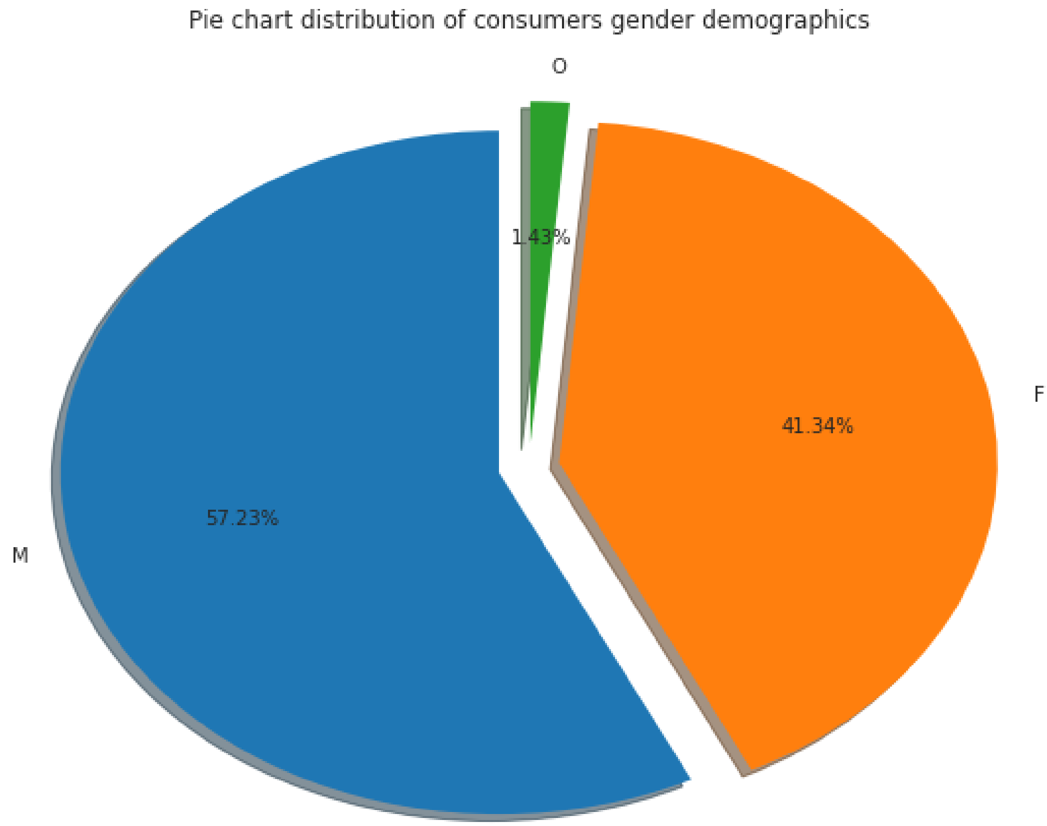
Below plot shows the distribution of consumer′s age

```
In [12]: sns.set_style("whitegrid")
         fig, ax = plt.subplots(figsize=(20,8))
         sns.distplot(a=profile['age'], color='g', kde = True);
         ax.set_title('Distribution of consumers age');
```




We can see abnormal amount of values near 120. This might be due to Null values replaced with this value. It will be delt in near future code.

```
In [13]: #pie plot of gender distribution
         fig, ax = plt.subplots(figsize=(10,8))
         plt.pie(profile['gender'].value_counts().values,labels= profile['gender'].value_counts(
         ax.set_title('Pie chart distribution of consumers gender demographics');
```



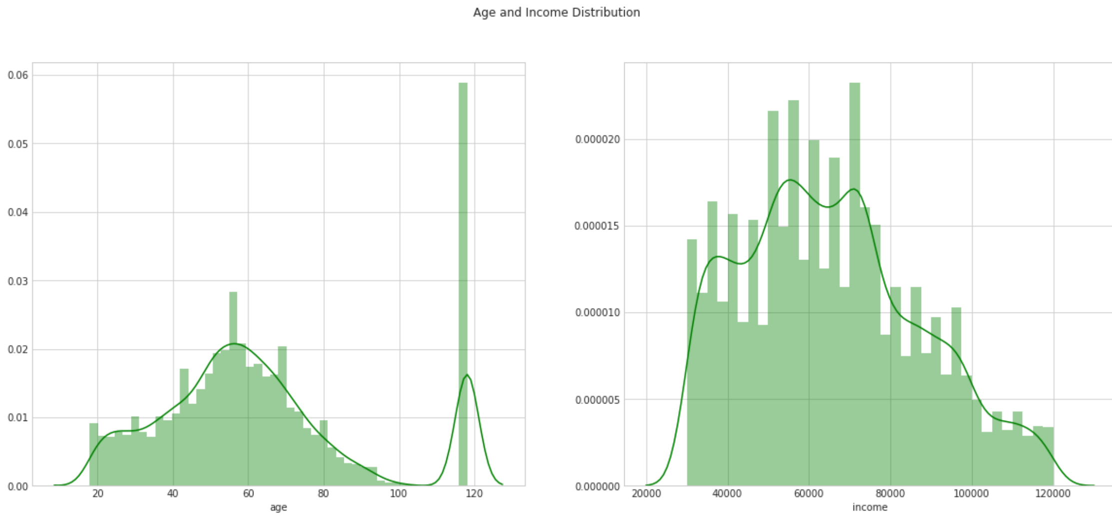

```
In [14]: profile[profile.isna().any(axis=1)].head()
```

Out[14]:

| | age | became_member_on | gender | id | income |
|---|---|---|---|---|---|
| 0 | 118 | 20170212 | None | 68be06ca386d4c31939f3a4f0e3dd783 | NaN |
| 2 | 118 | 20180712 | None | 38fe809add3b4fcf9315a9694bb96ff5 | NaN |
| 4 | 118 | 20170804 | None | a03223e636434f42ac4c3df47e8bac43 | NaN |
| 6 | 118 | 20170925 | None | 8ec6ce2a7e7949b1bf142def7d0e0586 | NaN |
| 7 | 118 | 20171002 | None | 68617ca6246f4fbc85e91a2a49552598 | NaN |

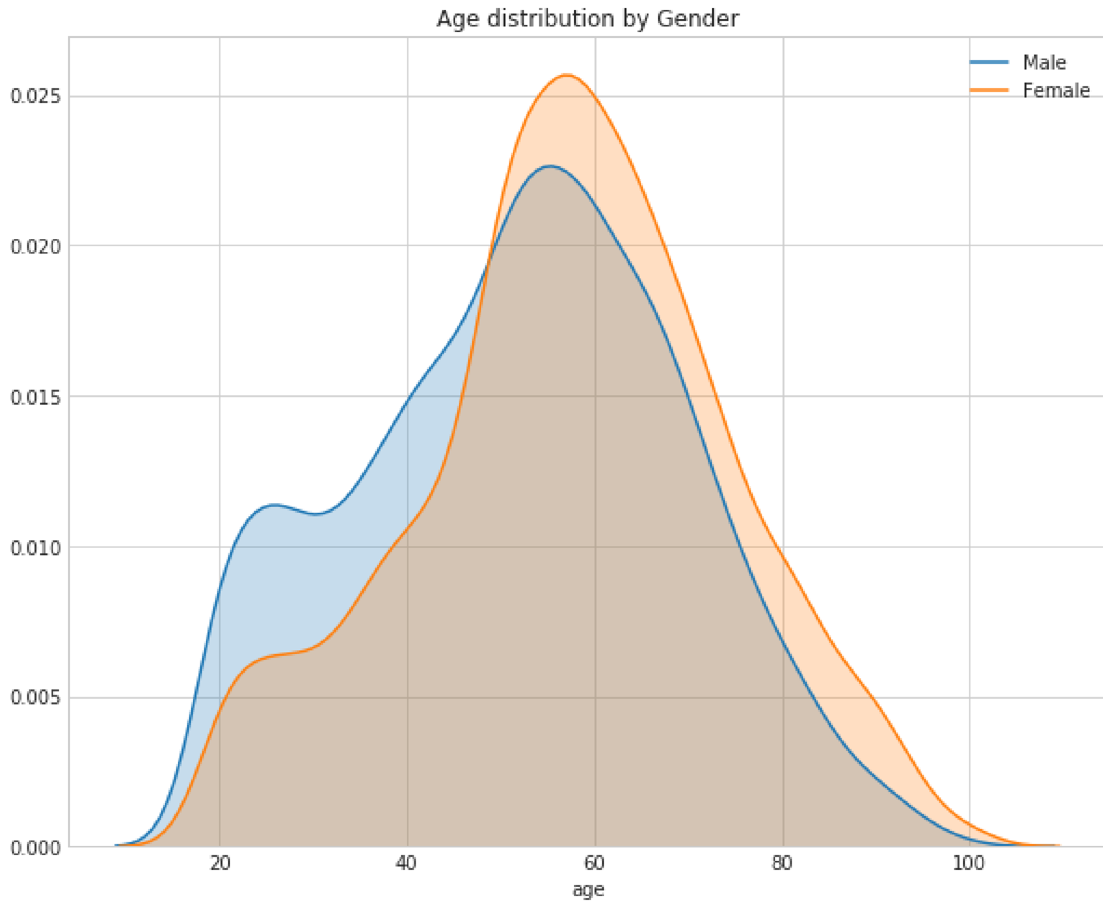
From above table we can confirm that Null values in age column is replaced with 118. Along with it None for gender column and **NaN** for income column

```
In [15]: #plotting age and income distribution
         fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20, 8))
         sns.distplot(a=profile['age'][profile['age'].notnull()], color='g', ax=axes[0]);
         sns.distplot(a=profile['income'][profile['income'].notnull()], color='g', ax=axes[1]);
         fig.suptitle('Age and Income Distribution');
```

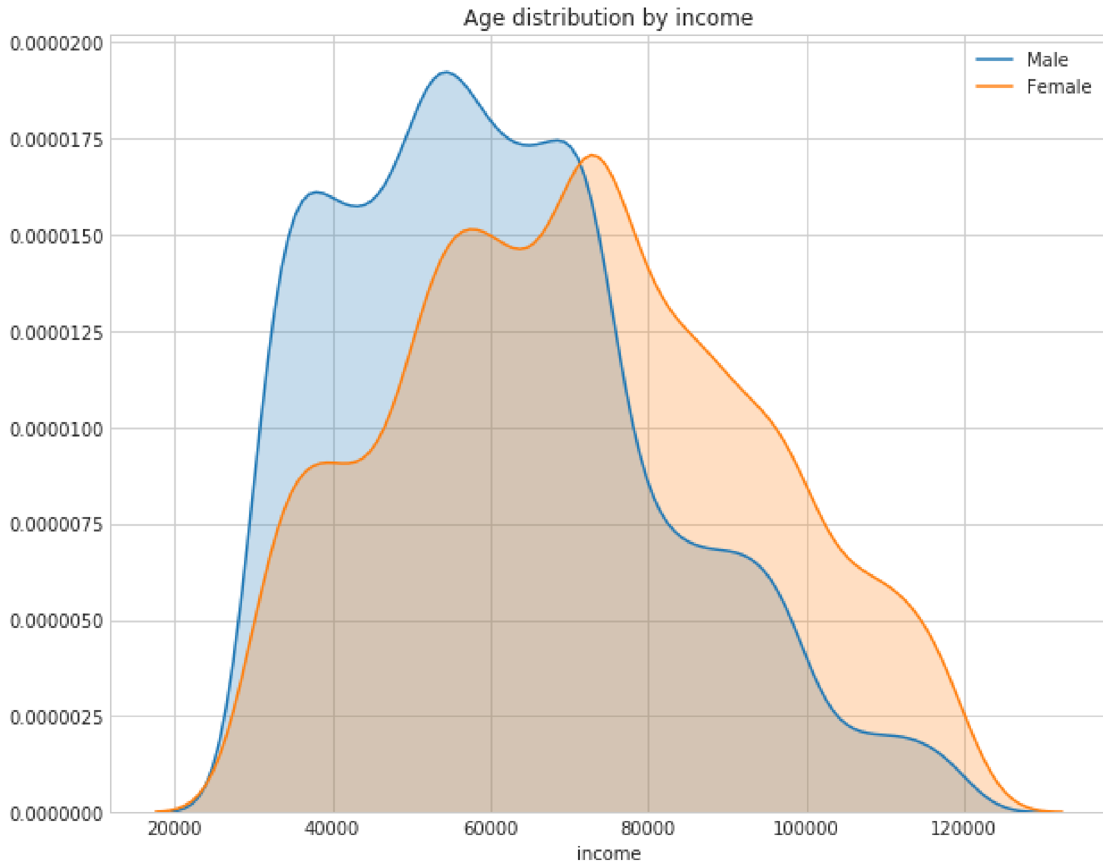



Above plots are made without Null values includes in it. From distribution it is clear: - Since the distribution of age is symetric the null values can be replaced by mean - Since the distribution of income is left skewed the null values can be replaced by median It will be delt in near future code.

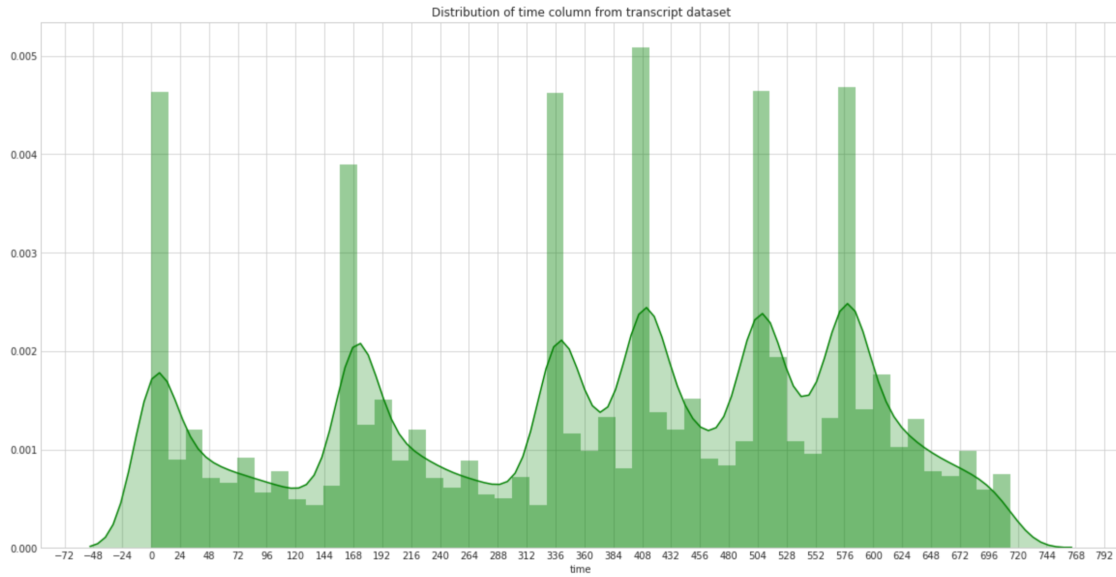
```
In [16]: #plotting age distribution by gender
         fig, ax = plt.subplots(figsize=(10,8));
         sns.distplot(profile[profile['gender']=='M']['age'],hist=False, kde_kws={"shade": True}
         sns.distplot(profile[profile['gender']=='F']['age'],hist=False, kde_kws={"shade": True}
         plt.title('Age distribution by Gender');
```




In [17]:
```
#plotting age distribution by income
fig, ax = plt.subplots(figsize=(10,8));
sns.distplot(profile[profile['gender']=='M']['income'],hist=False, kde_kws={"shade": Tr
sns.distplot(profile[profile['gender']=='F']['income'],hist=False,kde_kws={"shade": Tru
plt.title('Age distribution by income');
plt.legend();
```




```
In [18]: #plot distribution of time column
         fig, ax = plt.subplots(figsize=(20,10));
         sns.distplot(transcript['time'], color='g', hist = True, kde_kws={"shade": True});
         plt.title('Distribution of time column from transcript dataset');
         ax.xaxis.set_major_locator(ticker.MultipleLocator(24))
         ax.xaxis.set_major_formatter(ticker.ScalarFormatter())
```

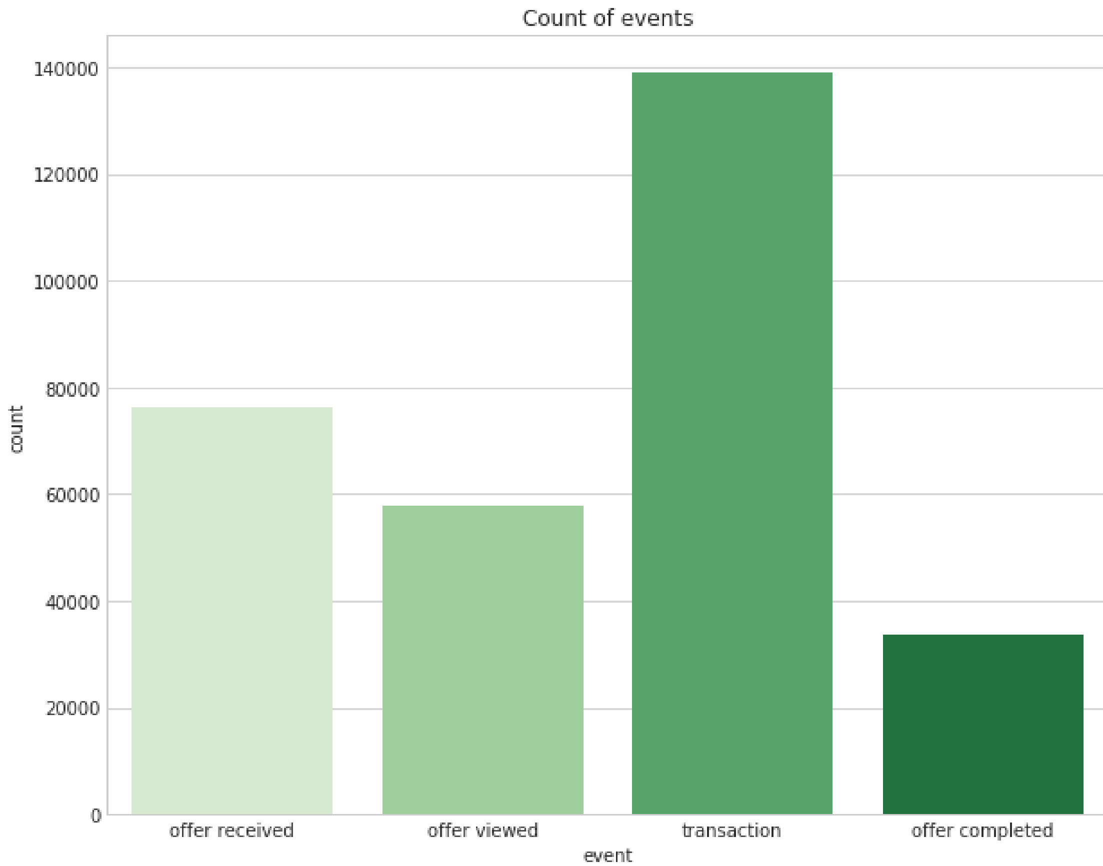



It is clear that distribution follows some pattern. From general knowledge, we know mornings will have high purchase volume from the stores as coffee is used boost the start of workday. Also it is also intutive that mondays will have high purchase volume then rest of the week.

Since no information is given about what and when the time = 0 starts from the transcript dataset. And since offer notifications, offer views drive purchases. **I will assume that time = 0 starts in the morning of Monday**

My assumptions can also be supported from the employees of starbuck's discussion: Link

In [19]:
```
#plot count of events
fig, ax = plt.subplots(figsize=(10,8));
sns.countplot(transcript['event'],palette="Greens");
plt.title('Count of events');
```




## 5 Preparing Data

Glancing at portfolio dataset..

In [20]:
```
portfolio.head()
```

Out[20]:
```
                      channels  difficulty  duration  \
0       [email, mobile, social]          10         7
1  [web, email, mobile, social]          10         5
2          [web, email, mobile]           0         4
3          [web, email, mobile]           5         7
4                  [web, email]          20        10

                                 id     offer_type  reward
0  ae264e3637204a6fb9bb56bc8210ddfd           bogo      10
1  4d5c57ea9a6940dd891ad53e9dbe8da0           bogo      10
2  3f207df678b143eea3cee63160fa8bed  informational       0
3  9b98b8c7a33c4b65b9aebfe6a799e6d9           bogo       5
4  0b1e1539f2cc45b7b9fa7c272da2e1d7       discount       5
```

We can see that channels is in list format. It cannot be used directly, so some preprocessing is required. Below code creates a new column and enters 1 if a particular channel is present or else 0

In [21]:
```
portfolio['channel_email'] = portfolio['channels'].apply(lambda x: 1 if 'email' in x el
portfolio['channel_mobile'] = portfolio['channels'].apply(lambda x: 1 if 'mobile' in x
portfolio['channel_social'] = portfolio['channels'].apply(lambda x: 1 if 'social' in x
portfolio['channel_web'] = portfolio['channels'].apply(lambda x: 1 if 'web' in x else 0
portfolio.drop(columns=['channels'], inplace=True)
```

In [22]:
```
portfolio
```

Out[22]:
```
   difficulty  duration                                id  offer_type  \
0          10         7  ae264e3637204a6fb9bb56bc8210ddfd          bogo
1          10         5  4d5c57ea9a6940dd891ad53e9dbe8da0          bogo
2           0         4  3f207df678b143eea3cee63160fa8bed  informational
3           5         7  9b98b8c7a33c4b65b9aebfe6a799e6d9          bogo
4          20        10  0b1e1539f2cc45b7b9fa7c272da2e1d7      discount
5           7         7  2298d6c36e964ae4a3e7e9706d1fb8c2      discount
6          10        10  fafdcd668e3743c1bb461111dcafc2a4      discount
7           0         3  5a8bc65990b245e5a138643cd4eb9837  informational
8           5         5  f19421c1d4aa40978ebb69ca19b0e20d          bogo
9          10         7  2906b810c7d4411798c6938adc9daaa5      discount

   reward  channel_email  channel_mobile  channel_social  channel_web
0      10              1               1               1            0
1      10              1               1               1            1
2       0              1               1               0            1
3       5              1               1               0            1
4       5              1               0               0            1
5       3              1               1               1            1
6       2              1               1               1            1
7       0              1               1               1            0
8       5              1               1               1            1
9       2              1               1               0            1
```

As per general knowledge, We know that we try to maximize reward given the difficulty. So I feel the ratio between reward and difficulty feature will help model in prediction. This process is called **Feature Engineering**

In [23]:
```
#create a column with reward/difficulty ratio and handle NaN values
portfolio['rew_by_diff'] = portfolio['reward']/portfolio['difficulty']
portfolio['rew_by_diff'].fillna(0, inplace=True)
```

As offer_type is categorical variable for future model prediction it is being encoded

In [24]:
```
#one-hot encoding
offers = pd.get_dummies(portfolio['offer_type'], prefix ='offer_type', prefix_sep='_')
```

In [25]:
```
#concatanate with original dataset
portfolio = pd.concat([portfolio, offers], axis=1)
```

```
In [26]: #remove original offer_type column
         portfolio.drop(columns=['offer_type'], inplace=True)
```

As id from portfolio can be confused with id from profile it is being renamed

```
In [27]: portfolio.rename(columns={'id':'offer_id'}, inplace=True)
```

Glancing at profile dataset..

```
In [28]: profile.head()
```

```
Out[28]:    age  became_member_on gender                                id    income
         0  118          20170212   None  68be06ca386d4c31939f3a4f0e3dd783       NaN
         1   55          20170715      F  0610b486422d4921ae7d2bf64640c50b  112000.0
         2  118          20180712   None  38fe809add3b4fcf9315a9694bb96ff5       NaN
         3   75          20170509      F  78afa995795e4d85b5d9ceeca43f5fef  100000.0
         4  118          20170804   None  a03223e636434f42ac4c3df47e8bac43       NaN
```

From earlier comment => age column 118 will be converted to nan and later it will filled using relevant imputation technique like mean in this case

```
In [29]: #convert 118 to np.nan
         profile['age'] = profile['age'].apply(lambda x: np.nan if x == 118 else x)
         #convert dtype of became_member_on column
         profile['became_member_on'] = pd.to_datetime(profile['became_member_on'],format='%Y%m%d
```

As became_member_on is in pandas datatime dtype but it cannot be used in analysis or in prediction. So it engineered so as to convert it into an numerical. which will still support same information.

```
In [30]: profile['became_member_on'] = (profile['became_member_on'] - profile['became_member_on'
```

As gender is categorical variable for future model prediction it is being encoded

```
In [31]: #one-hot encoding
         gender = pd.get_dummies(profile['gender'], prefix ='gender', prefix_sep='_')

In [32]: #concatenate with orginal dataset. Delete gender column
         profile = pd.concat([profile, gender], axis=1)
         profile.drop(columns=['gender'], inplace=True)

In [33]: #For clarity and merging which will be used in later part, The id column is renamed
         profile.rename(columns={'id':'consumer_id'}, inplace=True)

In [34]: #From above discussion we impute age column with mean
         profile['age'] = profile['age'].fillna(profile['age'].mean())

In [35]: #From above discussion we impute age column with median
         profile['income'] = profile['income'].fillna(profile['income'].median())
```

```
In [36]: #Function for categorization of age column for further analysis
         def age(x):
             if x <= 30:
                 return "Young_Adult"
             elif (x>30 and x<=60):
                 return "Adult"
             else:
                 return "Old"
```

```
In [37]: #Function for categorization of income column for further analysis
         def income(x):
             if x <= 50000:
                 return "Lower"
             elif (x > 50000 and x <= 90000):
                 return "Middle"
             else:
                 return "Upper"
```

```
In [38]: #apply the age function
         profile['Age_group'] = profile['age'].apply(age)
         #one hot encoding
         age_column = pd.get_dummies(profile['Age_group'], prefix ='Age_group', prefix_sep='_')
         #concatanate with original dataset
         profile = pd.concat([profile, age_column], axis=1)
         #drop the  Age_group column
         profile.drop(columns=['Age_group'], inplace=True)
```

```
In [39]: #Apply the income function
         profile['Income_group'] = profile['income'].apply(income)
         #one hot enoding
         income_column = pd.get_dummies(profile['Income_group'], prefix ='Income_group', prefix_
         #concatanate with orifinal dataset
         profile = pd.concat([profile, income_column], axis=1)
         #drop the Income_group column
         profile.drop(columns=['Income_group'], inplace=True)
```

```
In [40]: profile.head()
```

```
Out[40]:          age  became_member_on                       consumer_id    income  \
         0  54.393524          0.709819  68be06ca386d4c31939f3a4f0e3dd783   64000.0
         1  55.000000          0.793747  0610b486422d4921ae7d2bf64640c50b  112000.0
         2  54.393524          0.992320  38fe809add3b4fcf9315a9694bb96ff5   64000.0
         3  75.000000          0.756994  78afa995795e4d85b5d9ceeca43f5fef  100000.0
         4  54.393524          0.804717  a03223e636434f42ac4c3df47e8bac43   64000.0

            gender_F  gender_M  gender_O  Age_group_Adult  Age_group_Old  \
         0         0         0         0                1              0
         1         1         0         0                1              0
         2         0         0         0                1              0
```

```
3        1        0        0                 0              1
4        0        0        0                 1              0

   Age_group_Young_Adult  Income_group_Lower  Income_group_Middle  \
0                      0                   0                    1
1                      0                   0                    0
2                      0                   0                    1
3                      0                   0                    0
4                      0                   0                    1

   Income_group_Upper
0                   0
1                   1
2                   0
3                   1
4                   0
```

```
In [41]: profile.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 13 columns):
age                      17000 non-null float64
became_member_on         17000 non-null float64
consumer_id              17000 non-null object
income                   17000 non-null float64
gender_F                 17000 non-null uint8
gender_M                 17000 non-null uint8
gender_O                 17000 non-null uint8
Age_group_Adult          17000 non-null uint8
Age_group_Old            17000 non-null uint8
Age_group_Young_Adult    17000 non-null uint8
Income_group_Lower       17000 non-null uint8
Income_group_Middle      17000 non-null uint8
Income_group_Upper       17000 non-null uint8
dtypes: float64(3), object(1), uint8(9)
memory usage: 680.7+ KB
```

Glancing at transcript dataset..

```
In [42]: transcript.head()
```

```
Out[42]:             event                            person  time  \
         0  offer received  78afa995795e4d85b5d9ceeca43f5fef     0
         1  offer received  a03223e636434f42ac4c3df47e8bac43     0
         2  offer received  e2127556f4f64592b11af22de27a7932     0
         3  offer received  8ec6ce2a7e7949b1bf142def7d0e0586     0
         4  offer received  68617ca6246f4fbc85e91a2a49552598     0
```

```
                                            value
0  {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
1  {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
2  {'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3  {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4  {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}
```

In [43]:
```
#Function for categorization of time column for further analysis
def period(x):
    hour = x%24
    period_internal = int(hour/6)
    if period_internal < 1:
        return "Morning"
    elif (period_internal < 2) and (period_internal >= 1):
        return "Afternoon"
    elif (period_internal < 3) and (period_internal >= 2):
        return "Evening"
    else:
        return "Night"
```

In [44]:
```
#Function for categorization of time column for further analysis
def day(x):
    day_internal = int(x/24)
    if day_internal % 7 == 0:
        return "Monday"
    elif day_internal % 7 == 1:
        return "Tuesday"
    elif day_internal % 7 == 2:
        return "Wednesday"
    elif day_internal % 7 == 3:
        return "Thrusday"
    elif day_internal % 7 == 4:
        return "Friday"
    elif day_internal % 7 == 5:
        return "Saturday"
    else:
        return "Sunday"
```

In [45]:
```
#Apply period function
transcript['period'] = transcript['time'].apply(period)
#one hot encoding
period_column = pd.get_dummies(transcript['period'], prefix ='period', prefix_sep='_')
#concatanate with original dataset
transcript = pd.concat([transcript, period_column], axis=1)
#Will be used in later analysis
df_analysis_6 = transcript.copy()
```

```
#drop period column
transcript.drop(columns=['period'], inplace=True)
```

```
In [46]: #Apply day function
         transcript['day'] = transcript['time'].apply(day)
         #one hot encoding
         day_column = pd.get_dummies(transcript['day'], prefix ='day', prefix_sep='_')
         #concatanate with original dataset
         transcript = pd.concat([transcript, day_column], axis=1)
         #Will be used in later analysis
         df_analysis_4 = transcript.copy()
         #drop period column
         transcript.drop(columns=['day'], inplace=True)
```

```
In [47]: transcript.head()
```

```
Out[47]:             event                            person  time  \
         0  offer received  78afa995795e4d85b5d9ceeca43f5fef     0
         1  offer received  a03223e636434f42ac4c3df47e8bac43     0
         2  offer received  e2127556f4f64592b11af22de27a7932     0
         3  offer received  8ec6ce2a7e7949b1bf142def7d0e0586     0
         4  offer received  68617ca6246f4fbc85e91a2a49552598     0

                                                      value  period_Afternoon  \
         0  {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}                 0
         1  {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                 0
         2  {'offer id': '2906b810c7d4411798c6938adc9daaa5'}                 0
         3  {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}                 0
         4  {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}                 0

            period_Evening  period_Morning  period_Night  day_Friday  day_Monday  \
         0               0               1             0           0           1
         1               0               1             0           0           1
         2               0               1             0           0           1
         3               0               1             0           0           1
         4               0               1             0           0           1

            day_Saturday  day_Sunday  day_Thrusday  day_Tuesday  day_Wednesday
         0             0           0             0            0              0
         1             0           0             0            0              0
         2             0           0             0            0              0
         3             0           0             0            0              0
         4             0           0             0            0              0
```

Value column from this dataset is important as it contains the offer id, reward, amount according to the event. But preprocessing this column is tricky

```
In [48]: #As time column is measured in hours this transformation helps in finding trends in wee
         transcript['time'] = transcript['time']/24
```

```
transcript['offer_id'] = transcript['value'].apply(lambda x: x['offer_id'] if 'offer_id
num_vals = ['reward','amount']
for i in num_vals:
    transcript[i] = transcript['value'].apply(lambda x:x[i] if i in x else None)

transcript.drop('value',axis=1,inplace=True)
```

```
In [49]: #Average_frequency is engineered feature which will give us idea about how frequent ite
         #groupby person , event and apply difference of mean on time column and unstack with su
         average_frequency = transcript.groupby(by=['person', 'event'])['time'].apply(lambda x:
```

```
/opt/conda/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2889: RuntimeWarning: Mean of e
  out=out, **kwargs)
/opt/conda/lib/python3.6/site-packages/numpy/core/_methods.py:80: RuntimeWarning: invalid value
  ret = ret.dtype.type(ret / rcount)
```

```
In [50]: #merge it into original dataset
         transcript = transcript.merge(average_frequency, on = 'person')
```

```
In [51]: #counts is engineered feature which will give us idea about how many iteractions are ha
         #groupby person , event and apply count of time and unstack with suffix
         counts = transcript.groupby(by=['person', 'event'])['time'].count().unstack().add_suffi
         transcript = transcript.merge(counts, on = 'person')
```

```
In [52]: #average_amount is engineered feature which will give us idea about average amount spen
         #groupby person , amount and apply mean of amount and unstack with suffix
         average_amount = transcript.groupby(by=['person'])['amount'].mean().to_frame().rename(c
         transcript = transcript.merge(average_amount, on = 'person')
```

```
In [53]: #count_rewards is engineered feature which will give us idea about count of reward reci
         #groupby person , reward and apply mean of mean and unstack with suffix
         count_rewards = transcript.groupby(by=['person'])['reward'].count().to_frame().rename(c
         transcript = transcript.merge(count_rewards, on = 'person')
```

```
In [54]: #total_amount is engineered feature which will give us idea about total amount spent du
         #groupby person , amount and apply sum of amount and unstack with suffix
         total_amount = transcript.groupby(by=['person'])['amount'].sum().to_frame().rename(colu
         transcript = transcript.merge(total_amount, on = 'person')
```

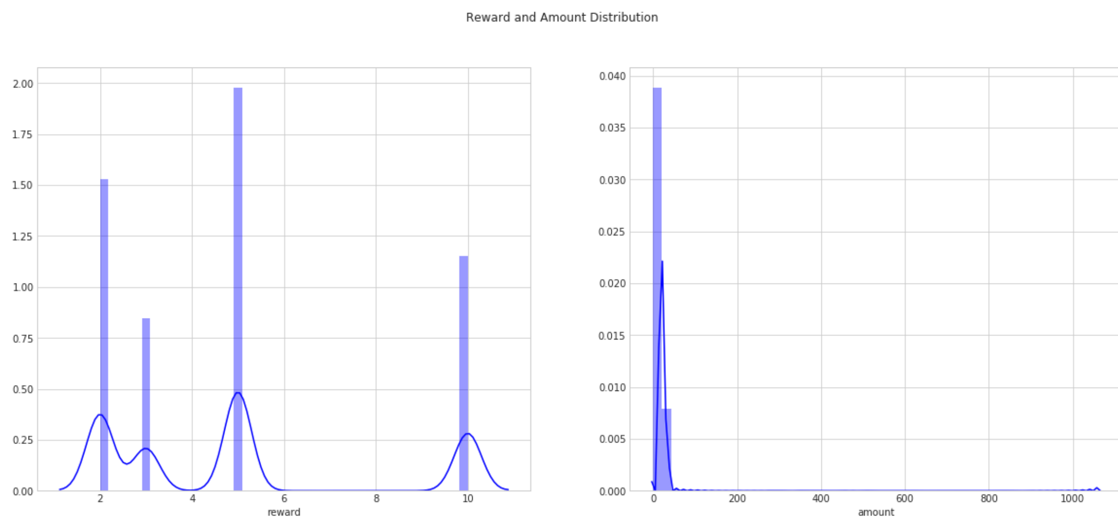
```
In [55]: transcript.rename(columns={'person': 'consumer_id'}, inplace=True)
```

```
In [56]: transcript.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 306534 entries, 0 to 306533
Data columns (total 28 columns):
event                        306534 non-null object
consumer_id                  306534 non-null object
```

```
time                          306534 non-null float64
period_Afternoon              306534 non-null uint8
period_Evening                306534 non-null uint8
period_Morning                306534 non-null uint8
period_Night                  306534 non-null uint8
day_Friday                    306534 non-null uint8
day_Monday                    306534 non-null uint8
day_Saturday                  306534 non-null uint8
day_Sunday                    306534 non-null uint8
day_Thrusday                  306534 non-null uint8
day_Tuesday                   306534 non-null uint8
day_Wednesday                 306534 non-null uint8
offer_id                      167581 non-null object
reward                        33579 non-null float64
amount                        138953 non-null float64
offer completed_frequency     205346 non-null float64
offer received_frequency      305948 non-null float64
offer viewed_frequency        294115 non-null float64
transaction_frequency         299454 non-null float64
offer completed_counts        254719 non-null float64
offer received_counts         306514 non-null float64
offer viewed_counts           305264 non-null float64
transaction_counts            303161 non-null float64
average_amount                303161 non-null float64
count_reward                  306534 non-null int64
total_amount                  306534 non-null float64
dtypes: float64(13), int64(1), object(3), uint8(11)
memory usage: 45.3+ MB
```

In [57]:
```
#plot reward and amount distribution
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20, 8))
sns.distplot(a=transcript['reward'][transcript['reward'].notnull()], color='b', ax=axes
sns.distplot(a=transcript['amount'][transcript['amount'].notnull()], color='b', ax=axes
fig.suptitle('Reward and Amount Distribution');
```




```
In [58]: transcript_later = transcript.copy()
```

```
In [59]: #drop reward and amount columns as relevant details are already captured
         transcript.drop(columns=['reward', 'amount'], inplace=True)
```

```
In [60]: transcript.fillna(transcript.median(), inplace=True)
```

**Merging all the dastset into a single master df**

```
In [61]: df = transcript.merge(portfolio, how='left', on='offer_id' )
         df = df.merge(profile, how='left', on='consumer_id')
```

```
In [62]: df.head()
```

```
Out[62]:              event                       consumer_id  time  period_Afternoon  \
         0   offer received  78afa995795e4d85b5d9ceeca43f5fef  0.00                 0
         1     offer viewed  78afa995795e4d85b5d9ceeca43f5fef  0.25                 1
         2      transaction  78afa995795e4d85b5d9ceeca43f5fef  5.50                 0
         3  offer completed  78afa995795e4d85b5d9ceeca43f5fef  5.50                 0
         4      transaction  78afa995795e4d85b5d9ceeca43f5fef  6.00                 0

            period_Evening  period_Morning  period_Night  day_Friday  day_Monday  \
         0               0               1             0           0           1
         1               0               0             0           0           1
         2               1               0             0           0           0
         3               1               0             0           0           0
         4               0               1             0           0           0

            day_Saturday  day_Sunday  day_Thrusday  day_Tuesday  day_Wednesday  \
         0             0           0             0            0              0
         1             0           0             0            0              0
```

```
2           1          0            0           0            0
3           1          0            0           0            0
4           0          1            0           0            0

                           offer_id  offer completed_frequency  \
0  9b98b8c7a33c4b65b9aebfe6a799e6d9                      7.875
1  9b98b8c7a33c4b65b9aebfe6a799e6d9                      7.875
2                              None                      7.875
3  9b98b8c7a33c4b65b9aebfe6a799e6d9                      7.875
4                              None                      7.875

   offer received_frequency  offer viewed_frequency  transaction_frequency  \
0                       7.0                     8.0               2.791667
1                       7.0                     8.0               2.791667
2                       7.0                     8.0               2.791667
3                       7.0                     8.0               2.791667
4                       7.0                     8.0               2.791667

   offer completed_counts  offer received_counts  offer viewed_counts  \
0                     3.0                    4.0                  4.0
1                     3.0                    4.0                  4.0
2                     3.0                    4.0                  4.0
3                     3.0                    4.0                  4.0
4                     3.0                    4.0                  4.0

   transaction_counts  average_amount  count_reward  total_amount  difficulty  \
0                 7.0       22.752857             3        159.27         5.0
1                 7.0       22.752857             3        159.27         5.0
2                 7.0       22.752857             3        159.27         NaN
3                 7.0       22.752857             3        159.27         5.0
4                 7.0       22.752857             3        159.27         NaN

   duration  reward  channel_email  channel_mobile  channel_social  \
0       7.0     5.0            1.0             1.0             0.0
1       7.0     5.0            1.0             1.0             0.0
2       NaN     NaN            NaN             NaN             NaN
3       7.0     5.0            1.0             1.0             0.0
4       NaN     NaN            NaN             NaN             NaN

   channel_web  rew_by_diff  offer_type_bogo  offer_type_discount  \
0          1.0          1.0              1.0                  0.0
1          1.0          1.0              1.0                  0.0
2          NaN          NaN              NaN                  NaN
3          1.0          1.0              1.0                  0.0
4          NaN          NaN              NaN                  NaN

   offer_type_informational   age  became_member_on    income  gender_F  \
0                       0.0  75.0          0.756994  100000.0         1
```

```
1                        0.0  75.0         0.756994  100000.0         1
2                        NaN  75.0         0.756994  100000.0         1
3                        0.0  75.0         0.756994  100000.0         1
4                        NaN  75.0         0.756994  100000.0         1

   gender_M  gender_O  Age_group_Adult  Age_group_Old  Age_group_Young_Adult  \
0         0         0                0              1                      0
1         0         0                0              1                      0
2         0         0                0              1                      0
3         0         0                0              1                      0
4         0         0                0              1                      0

   Income_group_Lower  Income_group_Middle  Income_group_Upper
0                   0                    0                   1
1                   0                    0                   1
2                   0                    0                   1
3                   0                    0                   1
4                   0                    0                   1
```

In [63]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 306534 entries, 0 to 306533
Data columns (total 49 columns):
event                        306534 non-null object
consumer_id                  306534 non-null object
time                         306534 non-null float64
period_Afternoon             306534 non-null uint8
period_Evening               306534 non-null uint8
period_Morning               306534 non-null uint8
period_Night                 306534 non-null uint8
day_Friday                   306534 non-null uint8
day_Monday                   306534 non-null uint8
day_Saturday                 306534 non-null uint8
day_Sunday                   306534 non-null uint8
day_Thrusday                 306534 non-null uint8
day_Tuesday                  306534 non-null uint8
day_Wednesday                306534 non-null uint8
offer_id                     167581 non-null object
offer completed_frequency    306534 non-null float64
offer received_frequency     306534 non-null float64
offer viewed_frequency       306534 non-null float64
transaction_frequency        306534 non-null float64
offer completed_counts       306534 non-null float64
offer received_counts        306534 non-null float64
offer viewed_counts          306534 non-null float64
transaction_counts           306534 non-null float64
average_amount               306534 non-null float64
```

```
count_reward                306534 non-null int64
total_amount                306534 non-null float64
difficulty                  167581 non-null float64
duration                    167581 non-null float64
reward                      167581 non-null float64
channel_email               167581 non-null float64
channel_mobile              167581 non-null float64
channel_social              167581 non-null float64
channel_web                 167581 non-null float64
rew_by_diff                 167581 non-null float64
offer_type_bogo             167581 non-null float64
offer_type_discount         167581 non-null float64
offer_type_informational    167581 non-null float64
age                         306534 non-null float64
became_member_on            306534 non-null float64
income                      306534 non-null float64
gender_F                    306534 non-null uint8
gender_M                    306534 non-null uint8
gender_O                    306534 non-null uint8
Age_group_Adult             306534 non-null uint8
Age_group_Old               306534 non-null uint8
Age_group_Young_Adult       306534 non-null uint8
Income_group_Lower          306534 non-null uint8
Income_group_Middle         306534 non-null uint8
Income_group_Upper          306534 non-null uint8
dtypes: float64(25), int64(1), object(3), uint8(20)
memory usage: 76.0+ MB
```

Now after merging. For analysis and plotting offer_id contains values which will difficult to interpret so we change into offer{Number}

```
In [64]: #list of unique offer_id
         unique_ids = list(df['offer_id'].unique())
         #saving unique string to each offer_id
         for i in range(len(unique_ids)):
             df['offer_id'] = df['offer_id'].apply(lambda x: f'offer{i+1}' if x == unique_ids[i]
```

```
In [65]: df_analysis_1 = df.copy()
```

As offer_id is categorical variable for future model prediction it is being encoded

```
In [66]: #one-hot encoding
         offer_ids = pd.get_dummies(df['offer_id'], prefix ='offer_id', prefix_sep='_')
         #concatanate with original dataset
         df = pd.concat([df, offer_ids], axis = 1)
         #drop offer_id column
         df.drop(columns=['offer_id'], inplace = True)
```

# 6 More Data Analysis

# 7 Which channel is effective in acheiveing offer views?

```
In [67]: #collect offer viewed data; offer received data along with channels from df
         df_analysis_1 = df[df['event'] == 'offer viewed'].iloc[:, 28:32].sum().to_frame().reset
         df_analysis_2 = df[df['event'] == 'offer received'].iloc[:, 28:32].sum().to_frame().res
         df_analysis_1.rename(columns = {0:'count_viewed', 'index': 'channel'}, inplace=True)
         df_analysis_2.rename(columns = {0:'count_received', 'index': 'channel'}, inplace=True)
         #merge both dataset
         df_analysis_3 = pd.merge(df_analysis_1, df_analysis_2, on = 'channel')
         df_analysis_3['Percent_Viewed'] = (df_analysis_3['count_viewed']/df_analysis_3['count_r
```

```
In [68]: df_analysis_3
```

```
Out[68]:          channel  count_viewed  count_received  Percent_Viewed
         0   channel_email       57725.0         76277.0       75.678121
         1  channel_mobile       55062.0         68609.0       80.254777
         2  channel_social       42629.0         45683.0       93.314800
         3     channel_web       44322.0         61001.0       72.657825
```

So it is clear that Social Media is most effective in acheiveing offer views

# 8 What are the best days for offer completions?

```
In [69]: df_analysis_4
```

```
Out[69]:                  event                            person  time  \
         0       offer received  78afa995795e4d85b5d9ceeca43f5fef     0
         1       offer received  a03223e636434f42ac4c3df47e8bac43     0
         2       offer received  e2127556f4f64592b11af22de27a7932     0
         3       offer received  8ec6ce2a7e7949b1bf142def7d0e0586     0
         4       offer received  68617ca6246f4fbc85e91a2a49552598     0
         5       offer received  389bc3fa690240e798340f5a15918d5c     0
         6       offer received  c4863c7985cf408faee930f111475da3     0
         7       offer received  2eeac8d8feae4a8cad5a6af0499a211d     0
         8       offer received  aa4862eba776480b8bb9c68455b8c2e1     0
         9       offer received  31dda685af34476cad5bc968bdb01c53     0
         10      offer received  744d603ef08c4f33af5a61c8c7628d1c     0
         11      offer received  3d02345581554e81b7b289ab5e288078     0
         12      offer received  4b0da7e80e5945209a1fdddfe813dbe0     0
         13      offer received  c27e0d6ab72c455a8bb66d980963de60     0
         14      offer received  d53717f5400c4e84affdaeda9dd926b3     0
         15      offer received  f806632c011441378d4646567f357a21     0
         16      offer received  d058f73bf8674a26a95227db098147b1     0
         17      offer received  65aba5c617294649aeb624da249e1ee5     0
         18      offer received  ebe7ef46ea6f4963a7dd49f501b26779     0
         19      offer received  1e9420836d554513ab90eba98552d0a9     0
```

```
20        offer received  868317b9be554cb18e50bc68484749a2     0
21        offer received  f082d80f0aac47a99173ba8ef8fc1909     0
22        offer received  102e9454054946fda62242d2e176fdce     0
23        offer received  4beeb3ed64dd4898b0edf2f6b67426d3     0
24        offer received  9f30b375d7bd4c62a884ffe7034e09ee     0
25        offer received  25c906289d154b66bf579693f89481c9     0
26        offer received  6e014185620b49bd98749f728747572f     0
27        offer received  02c083884c7d45b39cc68e1314fec56c     0
28        offer received  c0d210398dee4a0895b24444a5fcd1d2     0
29        offer received  8be4463721e14d7fa600686bf8c8b2ed     0
...                  ...                               ...   ...
306504       transaction  8524d450673b4c24869b6c94380006de   714
306505       transaction  b895c57e8cd047a8872ce02aa54759d6   714
306506   offer completed  b895c57e8cd047a8872ce02aa54759d6   714
306507      offer viewed  8dda575c2a1d44b9ac8e8b07b93d1f8e   714
306508       transaction  8431c16f8e1d440880db371a68f82dd0   714
306509   offer completed  8431c16f8e1d440880db371a68f82dd0   714
306510       transaction  ba620885e51c4b0ea64a4f61daad494f   714
306511       transaction  a1a8f40407c444cc848468275308958a   714
306512       transaction  8d80970192fa496f99d6b45c470a4b60   714
306513       transaction  bde275066f3c4fa0bff3093e3b866a2c   714
306514       transaction  f1e4fd36e5a0446f83861308bddf6945   714
306515       transaction  0b64be3b241c4407a5c9a71781173829   714
306516       transaction  86d03d35d7e0434b935e7743e83be3a0   714
306517       transaction  3408fd05c781401f8442fb6dbaaea9c7   714
306518       transaction  1593d617fac246ef8e50dbb0ffd77f5f   714
306519       transaction  f1b31d07b5d84f69a2d5f1d07843989e   714
306520       transaction  2ce987015ec0404a97ba333e8e814090   714
306521       transaction  2e33545f0a764d27b2ccff95fc8d72c4   714
306522       transaction  d1c4500ace2e45e9a45d3cd2fccac8d8   714
306523       transaction  b65affd9e07346a1906364a396950e3d   714
306524       transaction  d613ca9c59dd42f497bdbf6178da54a7   714
306525       transaction  eec70ab28af74a22a4aeb889c0317944   714
306526       transaction  24f56b5e1849462093931b164eb803b5   714
306527   offer completed  24f56b5e1849462093931b164eb803b5   714
306528       transaction  5ca2620962114246ab218fc648eb3934   714
306529       transaction  b3a1272bc9904337b331bf348c3e8c17   714
306530       transaction  68213b08d99a4ae1b0dcb72aebd9aa35   714
306531       transaction  a00058cf10334a308c68e7631c529907   714
306532       transaction  76ddbd6576844afe811f1a3c0fbb5bec   714
306533       transaction  c02b10e8752c4d8e9b73f918558531f7   714

                                                     value  period_Afternoon  \
0         {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}                 0
1         {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                 0
2         {'offer id': '2906b810c7d4411798c6938adc9daaa5'}                 0
3         {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}                 0
4         {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}                 0
```

```
5        {'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}                    0
6        {'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}                    0
7        {'offer id': '3f207df678b143eea3cee63160fa8bed'}                    0
8        {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                    0
9        {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                    0
10       {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                    0
11       {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                    0
12       {'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}                    0
13       {'offer id': '3f207df678b143eea3cee63160fa8bed'}                    0
14       {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                    0
15       {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}                    0
16       {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                    0
17       {'offer id': '2906b810c7d4411798c6938adc9daaa5'}                    0
18       {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}                    0
19       {'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}                    0
20       {'offer id': '2906b810c7d4411798c6938adc9daaa5'}                    0
21       {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}                    0
22       {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}                    0
23       {'offer id': '2906b810c7d4411798c6938adc9daaa5'}                    0
24       {'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}                    0
25       {'offer id': '2906b810c7d4411798c6938adc9daaa5'}                    0
26       {'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}                    0
27       {'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}                    0
28       {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}                    0
29       {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}                    0
...                                                    ...                 ...
306504                                  {'amount': 4.89}                    0
306505                                  {'amount': 4.48}                    0
306506  {'offer_id': 'fafdcd668e3743c1bb461111dcafc2a4...                  0
306507   {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}                   0
306508                                  {'amount': 1.19}                    0
306509  {'offer_id': 'fafdcd668e3743c1bb461111dcafc2a4...                  0
306510                                 {'amount': 14.31}                    0
306511                                  {'amount': 2.37}                    0
306512                                  {'amount': 6.92}                    0
306513                                 {'amount': 12.73}                    0
306514                                   {'amount': 8.2}                    0
306515                                   {'amount': 2.6}                    0
306516                                   {'amount': 9.2}                    0
306517                                  {'amount': 11.7}                    0
306518                                 {'amount': 40.67}                    0
306519                                 {'amount': 31.13}                    0
306520                     {'amount': 1.6400000000000001}                   0
306521                                 {'amount': 17.35}                    0
306522                                  {'amount': 4.42}                    0
306523                                 {'amount': 18.35}                    0
306524                                 {'amount': 25.14}                    0
306525                                 {'amount': 43.58}                    0
```

```
306526                               {'amount': 22.64}          0
306527  {'offer_id': 'fafdcd668e3743c1bb461111dcafc2a4...          0
306528                                 {'amount': 2.2}          0
306529                  {'amount': 1.5899999999999999}          0
306530                                {'amount': 9.53}          0
306531                                {'amount': 3.61}          0
306532                  {'amount': 3.5300000000000002}          0
306533                                {'amount': 4.05}          0

        period_Evening  period_Morning  period_Night      day  day_Friday  \
0                    0               1             0   Monday           0
1                    0               1             0   Monday           0
2                    0               1             0   Monday           0
3                    0               1             0   Monday           0
4                    0               1             0   Monday           0
5                    0               1             0   Monday           0
6                    0               1             0   Monday           0
7                    0               1             0   Monday           0
8                    0               1             0   Monday           0
9                    0               1             0   Monday           0
10                   0               1             0   Monday           0
11                   0               1             0   Monday           0
12                   0               1             0   Monday           0
13                   0               1             0   Monday           0
14                   0               1             0   Monday           0
15                   0               1             0   Monday           0
16                   0               1             0   Monday           0
17                   0               1             0   Monday           0
18                   0               1             0   Monday           0
19                   0               1             0   Monday           0
20                   0               1             0   Monday           0
21                   0               1             0   Monday           0
22                   0               1             0   Monday           0
23                   0               1             0   Monday           0
24                   0               1             0   Monday           0
25                   0               1             0   Monday           0
26                   0               1             0   Monday           0
27                   0               1             0   Monday           0
28                   0               1             0   Monday           0
29                   0               1             0   Monday           0
...                ...             ...           ...      ...         ...
306504               0               0             1  Tuesday           0
306505               0               0             1  Tuesday           0
306506               0               0             1  Tuesday           0
306507               0               0             1  Tuesday           0
306508               0               0             1  Tuesday           0
306509               0               0             1  Tuesday           0
306510               0               0             1  Tuesday           0
```

| | | | | | |
|---|---|---|---|---|---|
| 306511 | 0 | 0 | 1 | Tuesday | 0 |
| 306512 | 0 | 0 | 1 | Tuesday | 0 |
| 306513 | 0 | 0 | 1 | Tuesday | 0 |
| 306514 | 0 | 0 | 1 | Tuesday | 0 |
| 306515 | 0 | 0 | 1 | Tuesday | 0 |
| 306516 | 0 | 0 | 1 | Tuesday | 0 |
| 306517 | 0 | 0 | 1 | Tuesday | 0 |
| 306518 | 0 | 0 | 1 | Tuesday | 0 |
| 306519 | 0 | 0 | 1 | Tuesday | 0 |
| 306520 | 0 | 0 | 1 | Tuesday | 0 |
| 306521 | 0 | 0 | 1 | Tuesday | 0 |
| 306522 | 0 | 0 | 1 | Tuesday | 0 |
| 306523 | 0 | 0 | 1 | Tuesday | 0 |
| 306524 | 0 | 0 | 1 | Tuesday | 0 |
| 306525 | 0 | 0 | 1 | Tuesday | 0 |
| 306526 | 0 | 0 | 1 | Tuesday | 0 |
| 306527 | 0 | 0 | 1 | Tuesday | 0 |
| 306528 | 0 | 0 | 1 | Tuesday | 0 |
| 306529 | 0 | 0 | 1 | Tuesday | 0 |
| 306530 | 0 | 0 | 1 | Tuesday | 0 |
| 306531 | 0 | 0 | 1 | Tuesday | 0 |
| 306532 | 0 | 0 | 1 | Tuesday | 0 |
| 306533 | 0 | 0 | 1 | Tuesday | 0 |

| | day_Monday | day_Saturday | day_Sunday | day_Thrusday | day_Tuesday | \ |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 0 | 0 | 0 | 0 | |
| 2 | 1 | 0 | 0 | 0 | 0 | |
| 3 | 1 | 0 | 0 | 0 | 0 | |
| 4 | 1 | 0 | 0 | 0 | 0 | |
| 5 | 1 | 0 | 0 | 0 | 0 | |
| 6 | 1 | 0 | 0 | 0 | 0 | |
| 7 | 1 | 0 | 0 | 0 | 0 | |
| 8 | 1 | 0 | 0 | 0 | 0 | |
| 9 | 1 | 0 | 0 | 0 | 0 | |
| 10 | 1 | 0 | 0 | 0 | 0 | |
| 11 | 1 | 0 | 0 | 0 | 0 | |
| 12 | 1 | 0 | 0 | 0 | 0 | |
| 13 | 1 | 0 | 0 | 0 | 0 | |
| 14 | 1 | 0 | 0 | 0 | 0 | |
| 15 | 1 | 0 | 0 | 0 | 0 | |
| 16 | 1 | 0 | 0 | 0 | 0 | |
| 17 | 1 | 0 | 0 | 0 | 0 | |
| 18 | 1 | 0 | 0 | 0 | 0 | |
| 19 | 1 | 0 | 0 | 0 | 0 | |
| 20 | 1 | 0 | 0 | 0 | 0 | |
| 21 | 1 | 0 | 0 | 0 | 0 | |
| 22 | 1 | 0 | 0 | 0 | 0 | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 23 | 1 | 0 | 0 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 26 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 28 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... |
| 306504 | 0 | 0 | 0 | 0 | 1 |
| 306505 | 0 | 0 | 0 | 0 | 1 |
| 306506 | 0 | 0 | 0 | 0 | 1 |
| 306507 | 0 | 0 | 0 | 0 | 1 |
| 306508 | 0 | 0 | 0 | 0 | 1 |
| 306509 | 0 | 0 | 0 | 0 | 1 |
| 306510 | 0 | 0 | 0 | 0 | 1 |
| 306511 | 0 | 0 | 0 | 0 | 1 |
| 306512 | 0 | 0 | 0 | 0 | 1 |
| 306513 | 0 | 0 | 0 | 0 | 1 |
| 306514 | 0 | 0 | 0 | 0 | 1 |
| 306515 | 0 | 0 | 0 | 0 | 1 |
| 306516 | 0 | 0 | 0 | 0 | 1 |
| 306517 | 0 | 0 | 0 | 0 | 1 |
| 306518 | 0 | 0 | 0 | 0 | 1 |
| 306519 | 0 | 0 | 0 | 0 | 1 |
| 306520 | 0 | 0 | 0 | 0 | 1 |
| 306521 | 0 | 0 | 0 | 0 | 1 |
| 306522 | 0 | 0 | 0 | 0 | 1 |
| 306523 | 0 | 0 | 0 | 0 | 1 |
| 306524 | 0 | 0 | 0 | 0 | 1 |
| 306525 | 0 | 0 | 0 | 0 | 1 |
| 306526 | 0 | 0 | 0 | 0 | 1 |
| 306527 | 0 | 0 | 0 | 0 | 1 |
| 306528 | 0 | 0 | 0 | 0 | 1 |
| 306529 | 0 | 0 | 0 | 0 | 1 |
| 306530 | 0 | 0 | 0 | 0 | 1 |
| 306531 | 0 | 0 | 0 | 0 | 1 |
| 306532 | 0 | 0 | 0 | 0 | 1 |
| 306533 | 0 | 0 | 0 | 0 | 1 |

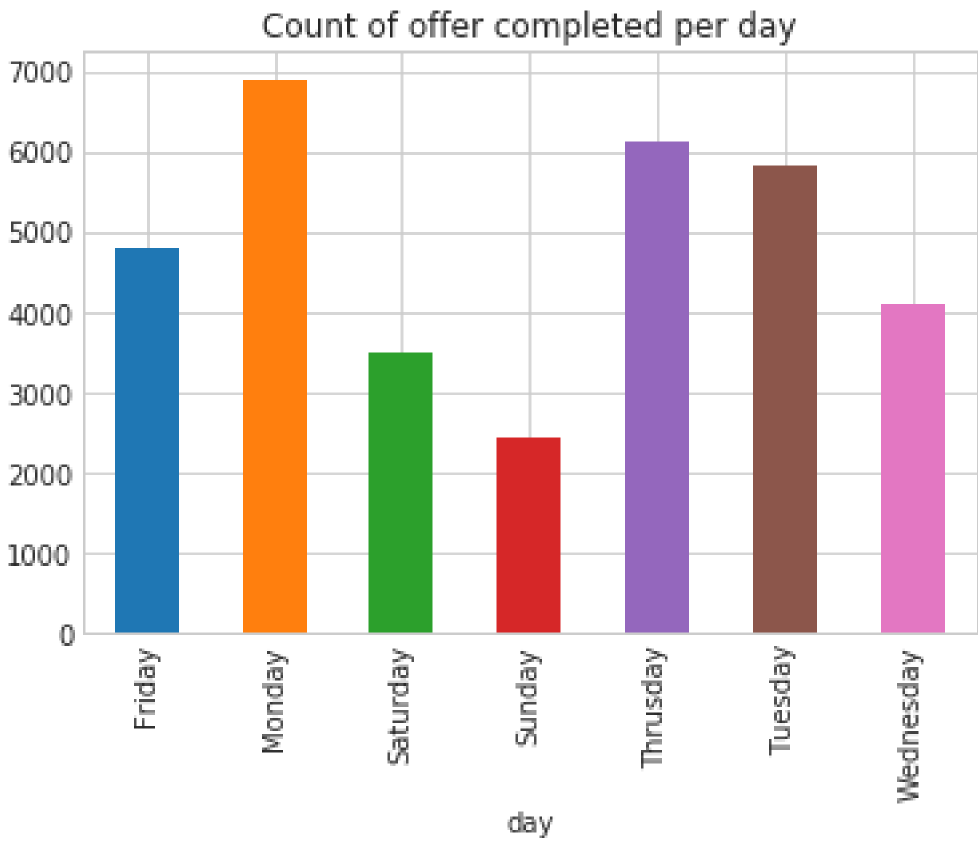
|  | day_Wednesday |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |

| | |
|---|---|
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |
| 26 | 0 |
| 27 | 0 |
| 28 | 0 |
| 29 | 0 |
| ... | ... |
| 306504 | 0 |
| 306505 | 0 |
| 306506 | 0 |
| 306507 | 0 |
| 306508 | 0 |
| 306509 | 0 |
| 306510 | 0 |
| 306511 | 0 |
| 306512 | 0 |
| 306513 | 0 |
| 306514 | 0 |
| 306515 | 0 |
| 306516 | 0 |
| 306517 | 0 |
| 306518 | 0 |
| 306519 | 0 |
| 306520 | 0 |
| 306521 | 0 |
| 306522 | 0 |
| 306523 | 0 |
| 306524 | 0 |
| 306525 | 0 |
| 306526 | 0 |
| 306527 | 0 |
| 306528 | 0 |

```
306529              0
306530              0
306531              0
306532              0
306533              0

[306534 rows x 16 columns]
```

In [70]:
```
#groupby day and count events
df_analysis_5 = df_analysis_4[df_analysis_4['event'] == 'offer completed'].groupby(by=[
```

In [71]:
```
df_analysis_5.plot(kind='bar');
plt.title('Count of offer completed per day');
```

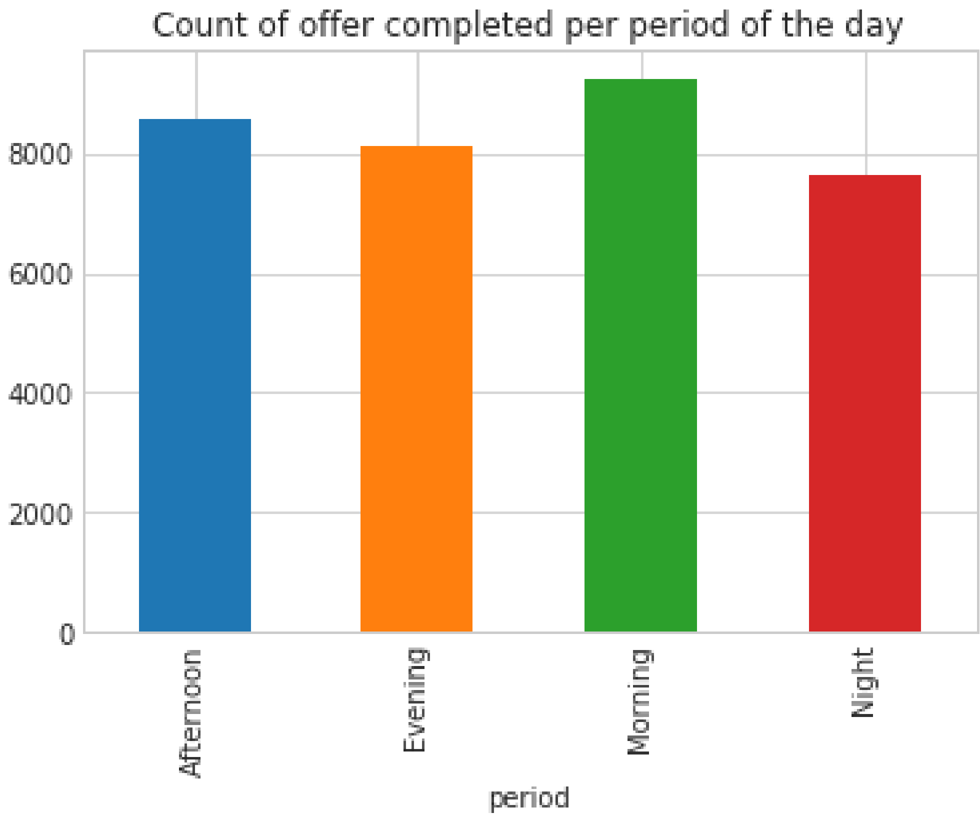



It is clear that Mondays are best followed by Thrusay and Tuesday

# 9 In what period is most offers completed?

In [72]:
```
#groupby period and count event
df_analysis_7 = df_analysis_6[df_analysis_6['event'] == 'offer completed'].groupby(by=[
```

```
In [73]: df_analysis_7.plot(kind='bar');
         plt.title('Count of offer completed per period of the day');
```




It is clear that Mornings have most offer completion

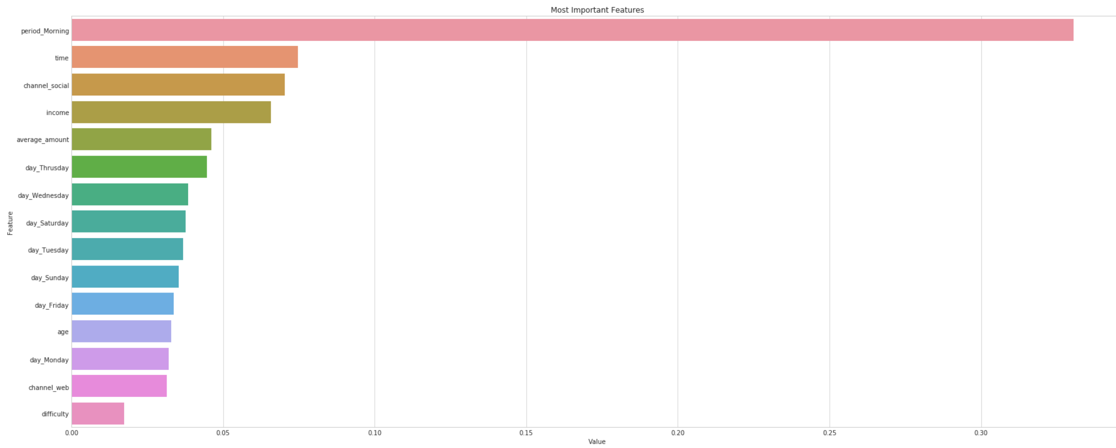
# 10 Predictive Model

# 11 What features contribute most for viewing the offer?

```
In [74]: df1 = df[((df['event']=='offer received')| (df['event']=='offer viewed'))]

In [75]: #drop the columsn with no use and with directly contains information of target variable
         X1 = df1.drop(columns=['event', 'consumer_id', 'offer viewed_counts', 'offer viewed_fre
         Y1 = df1['event'].apply(lambda x: 0 if (x == 'offer received') else 1).reset_index()['e

In [76]: X1.shape

Out[76]: (134002, 55)
```

Below function splits the data into test and train. Graident Boosting Classifier is used because because of its efficacy in dealing with large complex tabular datasets. Finally feature importance will be plotted and model will be returned

```
In [77]: def feature_importance(X, Y):
             '''
             Plots important features and return the machine learning model
             Input:
             X: Dataframe containing the input columns
             Y: Series containing the output column
             Ouput:
             Returns model fitted on Input data
             '''
             #Split the data
             X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_s
             #initiate the model
             model = GradientBoostingClassifier()
             #Cross Validation
             kfold = KFold(n_splits=5)
             cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='roc_auc',
             print("Mean: %f ;standard deviation: %f" % (cv_results.mean(), cv_results.std()))
             #fit the data
             model.fit(X_train, Y_train)
             #predict the the test data
             predictions = model.predict(X_test)
             print("Accuracy: %f" % accuracy_score(predictions ,Y_test))
             print("AUC : %f" % roc_auc_score(predictions ,Y_test))
             #store feature importance
             feature_imp = pd.DataFrame(sorted(zip(model.feature_importances_,X_train.columns)),
             plt.figure(figsize=(25, 10))
             sns.barplot(x="Value", y="Feature", data=feature_imp.sort_values(by="Value", ascend
             plt.tight_layout()
             plt.show()
             return model

In [78]: model1 = feature_importance(X1, Y1)

Mean: 0.923400 ;standard deviation: 0.001714
Accuracy: 0.903586
AUC : 0.927660
```




Above Visulization have huge benfit. Following are the insights that can be derived: - Morning period play important role for offer viewing - Social Media advertisement is highly effective in offer viewing - Income and Averge amount spend play role in viewing. - Next the day of the week has huge impact on whether the offer is viewed and left alone. This will help in targeting most important offers on particular days

## 12 What features contribute most for completing the offer?

In [79]:
```
df2 = df[((df['event']=='offer viewed')| (df['event']=='offer completed') | (df['event'
```

In [80]:
```
df2.head()
```

Out[80]:
```
             event                       consumer_id  time  period_Afternoon  \
0   offer received  78afa995795e4d85b5d9ceeca43f5fef  0.00                 0
1     offer viewed  78afa995795e4d85b5d9ceeca43f5fef  0.25                 1
3  offer completed  78afa995795e4d85b5d9ceeca43f5fef  5.50                 0
5   offer received  78afa995795e4d85b5d9ceeca43f5fef  7.00                 0
6     offer viewed  78afa995795e4d85b5d9ceeca43f5fef  9.00                 0

   period_Evening  period_Morning  period_Night  day_Friday  day_Monday  \
0               0               1             0           0           1
1               0               0             0           0           1
3               1               0             0           0           0
5               0               1             0           0           1
6               0               1             0           0           0

   day_Saturday  day_Sunday  day_Thrusday  day_Tuesday  day_Wednesday  \
0             0           0             0            0              0
1             0           0             0            0              0
3             1           0             0            0              0
5             0           0             0            0              0
6             0           0             0            0              1
```

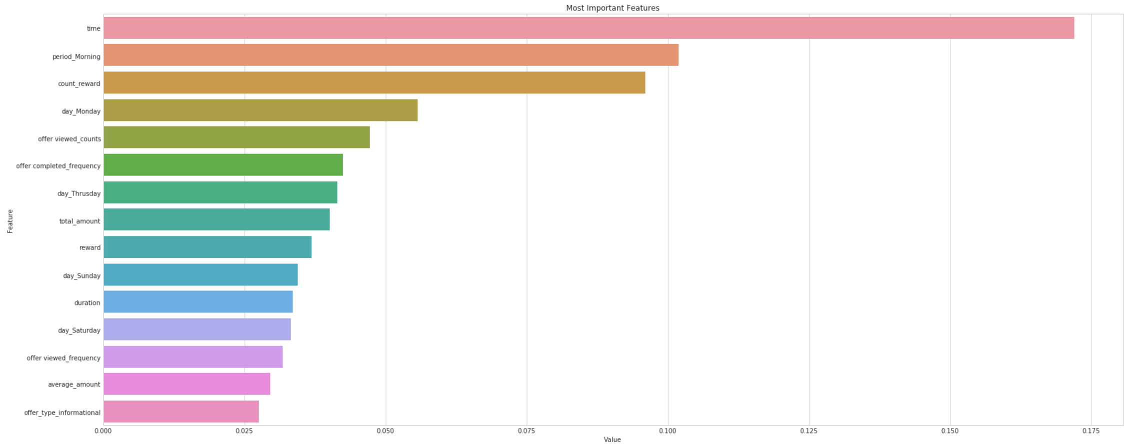
```
   offer completed_frequency  offer received_frequency  \
0                      7.875                       7.0
1                      7.875                       7.0
3                      7.875                       7.0
5                      7.875                       7.0
6                      7.875                       7.0

   offer viewed_frequency  transaction_frequency  offer completed_counts  \
0                     8.0               2.791667                     3.0
1                     8.0               2.791667                     3.0
3                     8.0               2.791667                     3.0
5                     8.0               2.791667                     3.0
6                     8.0               2.791667                     3.0

   offer received_counts  offer viewed_counts  transaction_counts  \
0                    4.0                  4.0                 7.0
1                    4.0                  4.0                 7.0
3                    4.0                  4.0                 7.0
5                    4.0                  4.0                 7.0
6                    4.0                  4.0                 7.0

   average_amount  count_reward  total_amount  difficulty  duration  reward  \
0       22.752857             3        159.27         5.0       7.0     5.0
1       22.752857             3        159.27         5.0       7.0     5.0
3       22.752857             3        159.27         5.0       7.0     5.0
5       22.752857             3        159.27         0.0       3.0     0.0
6       22.752857             3        159.27         0.0       3.0     0.0

   channel_email  channel_mobile  channel_social  channel_web  rew_by_diff  \
0            1.0             1.0             0.0          1.0          1.0
1            1.0             1.0             0.0          1.0          1.0
3            1.0             1.0             0.0          1.0          1.0
5            1.0             1.0             1.0          0.0          0.0
6            1.0             1.0             1.0          0.0          0.0

   offer_type_bogo  offer_type_discount  offer_type_informational   age  \
0              1.0                  0.0                       0.0  75.0
1              1.0                  0.0                       0.0  75.0
3              1.0                  0.0                       0.0  75.0
5              0.0                  0.0                       1.0  75.0
6              0.0                  0.0                       1.0  75.0

   became_member_on    income  gender_F  gender_M  gender_O  Age_group_Adult  \
0          0.756994  100000.0         1         0         0                0
1          0.756994  100000.0         1         0         0                0
3          0.756994  100000.0         1         0         0                0
5          0.756994  100000.0         1         0         0                0
```

```
6          0.756994  100000.0         1         0         0                0

   Age_group_Old  Age_group_Young_Adult  Income_group_Lower  \
0              1                      0                   0
1              1                      0                   0
3              1                      0                   0
5              1                      0                   0
6              1                      0                   0

   Income_group_Middle  Income_group_Upper  offer_id_offer1  offer_id_offer10  \
0                    0                   1                1                 0
1                    0                   1                1                 0
3                    0                   1                1                 0
5                    0                   1                0                 0
6                    0                   1                0                 0

   offer_id_offer11  offer_id_offer2  offer_id_offer3  offer_id_offer4  \
0                 0                0                0                0
1                 0                0                0                0
3                 0                0                0                0
5                 0                0                1                0
6                 0                0                1                0

   offer_id_offer5  offer_id_offer6  offer_id_offer7  offer_id_offer8  \
0                0                0                0                0
1                0                0                0                0
3                0                0                0                0
5                0                0                0                0
6                0                0                0                0

   offer_id_offer9
0                0
1                0
3                0
5                0
6                0
```

```
In [81]: #drop the columsn with no use and with directly contains information of target variable
         X2 = df2.drop(columns=['event', 'consumer_id'])
         Y2 = df2['event'].apply(lambda x: 1 if (x == 'offer completed') else 0).reset_index()['
```

```
In [82]: model2 = feature_importance(X2, Y2)
```

```
Mean: 0.912877 ;standard deviation: 0.001245
Accuracy: 0.863055
AUC : 0.790204
```




Following insights can be derived from above plot: - Just like in the case of offer viewing, Morning period play important role in offer completion - Next, Looking at count reward and offer_completed_count indicates that people keep buying as they recieve rewards and become loyal customers - It is suprising to offer viewed count and offer viewed frequency indicating that personalized frequent offers sent will impact offer completion - Thrusday is also important day for offer completion

# 13 Evaluation

Results are showcased in the medium blog post link: https://anudeeppeela9.medium.com/starbucks-capstone-challenge-e901baeff5d2

#### 13.0.1 Refinement Documentation

```
In [85]: #Hyperparmeter tuning
         def tuning(X, Y):
             '''
             Prints the score after tuning and return the tuned model
             Input:
             X: Dataframe containing the input columns
             Y: Series containing the output column
             Ouput:
             Returns model fitted on Input data
             '''
             #Split the data
             X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_s
             #Hyperparameters
             num_estimators = [250, 500]
             learn_rates = [0.05, 0.1]
             #initiate the model
             model = GradientBoostingClassifier()
             #parameter grid
```

```
            param_grid = {'n_estimators': num_estimators, 'learning_rate': learn_rates}
            kfold = KFold(n_splits=3)
            #gridsearchcv
            grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring='roc_auc', cv=k
            grid_result = grid.fit(X_train, Y_train)
            print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
            #predict the the test data
            predictions = grid.predict(X_test)
            print("Accuracy: %f" % accuracy_score(predictions ,Y_test))
            print("AUC : %f" % roc_auc_score(predictions ,Y_test))
            return model
```

In [86]:
```
tuning(X1, Y1)
```

```
Best: 0.923374 using {'learning_rate': 0.05, 'n_estimators': 250}
Accuracy: 0.903884
AUC : 0.927851
```

Out[86]:
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False)
```

In [87]:
```
tuning(X2, Y2)
```

```
Best: 0.917053 using {'learning_rate': 0.1, 'n_estimators': 500}
Accuracy: 0.870364
AUC : 0.797834
```

Out[87]:
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False)
```

In [88]:
```
#logistic regression
from sklearn.linear_model import LogisticRegression
def feature_importance_1(X, Y):
    '''
    Prints accuray and outputs model
```

```
    Input:
    X: Dataframe containing the input columns
    Y: Series containing the output column
    Ouput:
    Returns model fitted on Input data
    '''
    #Split the data
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_s
    #initiate the model
    model = LogisticRegression()
    #fit the data
    model.fit(X_train, Y_train)
    #predict the the test data
    predictions = model.predict(X_test)
    print("Accuracy: %f" % accuracy_score(predictions ,Y_test))
    print("AUC : %f" % roc_auc_score(predictions ,Y_test))
    return model
```

In [89]:
```
feature_importance_1(X1, Y1)
```

```
Accuracy: 0.872094
AUC : 0.908228
```

Out[89]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [90]:
```
feature_importance_1(X2, Y2)
```

```
Accuracy: 0.797446
AUC : 0.580042
```

Out[90]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [92]:
```
!pip install lightgbm
```

```
Collecting lightgbm
  Downloading https://files.pythonhosted.org/packages/18/b2/fff8370f48549ce223f929fe8cab4ee6bf28
    100% || 2.0MB 8.3MB/s eta 0:00:01
Requirement already satisfied: wheel in /opt/conda/lib/python3.6/site-packages (from lightgbm) (
Requirement already satisfied: numpy in /opt/conda/lib/python3.6/site-packages (from lightgbm) (
Requirement already satisfied: scikit-learn!=0.22.0 in /opt/conda/lib/python3.6/site-packages (f
Requirement already satisfied: scipy in /opt/conda/lib/python3.6/site-packages (from lightgbm) (
```

```
Installing collected packages: lightgbm
Successfully installed lightgbm-3.2.1
```

```
In [93]: #light lgb
         import lightgbm as lgb
         def feature_importance_2(X, Y):
             '''
             Returns model fitted on Input data
             Input:
             X: Dataframe containing the input columns
             Y: Series containing the output column
             Ouput:
             Returns model fitted on Input data
             '''
             #Split the data
             X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_s
             #initiate the model
             model = lgb.LGBMClassifier()
             #fit the data
             model.fit(X_train, Y_train)
             #predict the the test data
             predictions = model.predict(X_test)
             print("Accuracy: %f" % accuracy_score(predictions ,Y_test))
             print("AUC : %f" % roc_auc_score(predictions ,Y_test))
             return model
```

```
In [94]: feature_importance_2(X1, Y1)
```

```
Accuracy: 0.903847
AUC : 0.927791
```

```
Out[94]: LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
                        importance_type='split', learning_rate=0.1, max_depth=-1,
                        min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
                        n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
                        random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
                        subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

```
In [95]: feature_importance_2(X2, Y2)
```

```
Accuracy: 0.870066
AUC : 0.796191
```

```
Out[95]: LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
                        importance_type='split', learning_rate=0.1, max_depth=-1,
                        min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
```

```
               n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

```
In [ ]:
```