Revenue Prediction of RED

Background: RED is a lifestyle platform and a consumer decision-making portal. It is currently a well-known e-commerce platform. In RED community, users share their lives through text, pictures, and video notes to record life. In October 2014, RED Welfare Society came to solve another problem of overseas shopping. Through the machine learning, RED matches massive information and people accurately and efficiently. It has accumulated a large amount of overseas shopping data, analyzed the most popular products and global shopping trends, and based on this, the good things in the world are provided to the user by the shortest path and the simplest way. This assignment is based on the data of RED, using Python for linear regression modeling and analysis.

Analysis Steps:

1. Check the data dictionary and know the meaning of each field
2. Check the data info and clean the data
3. Correlation analysis among the variables and visualization
4. Data modeling
5. Model assessment and optimization

Attached the coding screen shot:

In [1]:
```
#import package needed
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
red=pd.read_csv('12_week2.csv')
```

Data Cleaning

In [3]:
```
red.info()#check the data info,found fields:gender,age,engaged_last_30 got null values, lifecycle's type is object.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29452 entries, 0 to 29451
Data columns (total 8 columns):
revenue                  29452 non-null float64
gender                   17723 non-null float64
age                      16716 non-null float64
engaged_last_30          17723 non-null float64
lifecycle                29452 non-null object
days_since_last_order    29452 non-null float64
previous_order_amount    29452 non-null float64
3rd_party_stores         29452 non-null int64
dtypes: float64(6), int64(1), object(1)
memory usage: 1.8+ MB
```

In [4]:
```
red.describe()#check the numeric type variables,found max age is 99,it's a little strange,maybe take some actions later.
```
Out[4]:

	revenue	gender	age	engaged_last_30	days_since_last_order	previous_order_amount	3rd_party_stores
count	29452.000000	17723.000000	16716.000000	17723.000000	29452.000000	29452.000000	29452.000000
mean	398.288037	0.950742	60.397404	0.073069	7.711348	2348.904830	2.286059
std	960.251728	0.216412	14.823026	0.260257	6.489289	2379.774213	3.538219
min	0.020000	0.000000	18.000000	0.000000	0.130000	0.000000	0.000000
25%	74.970000	1.000000	50.000000	0.000000	2.190000	773.506250	0.000000
50%	175.980000	1.000000	60.000000	0.000000	5.970000	1655.980000	0.000000
75%	499.990000	1.000000	70.000000	0.000000	11.740000	3096.766500	3.000000
max	103466.100000	1.000000	99.000000	1.000000	23.710000	11597.900000	10.000000

In [5]:
```
red.lifecycle.value_counts()#check lifecycle type
```
Out[5]:
```
C    20201
B     5709
A     3542
Name: lifecycle, dtype: int64
```

In [6]:
```
red.head()
```
Out[6]:

	revenue	gender	age	engaged_last_30	lifecycle	days_since_last_order	previous_order_amount	3rd_party_stores
0	72.98	1.0	59.0	0.0	B	4.26	2343.870	0
1	200.99	1.0	51.0	0.0	A	0.94	8539.872	0
2	69.98	1.0	79.0	0.0	C	4.29	1687.646	1
3	649.99	NaN	NaN	NaN	C	14.90	3498.846	0
4	83.59	NaN	NaN	NaN	C	21.13	3968.490	4

In [7]:
```
red.gender.fillna('unknown',inplace=True)#fill NaN in gender with unknown
```

In [8]:
```
red.engaged_last_30.fillna('unknown',inplace=True)#fill NaN in engaged_last_30 with unknown
```

In [9]:
```
red.age.fillna(red.age.mean(),inplace=True)#fill NaN in age with average age
```

In [10]:
```
red.info()#There are 3 object fields,need to convert.
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29452 entries, 0 to 29451
Data columns (total 8 columns):
revenue                  29452 non-null float64
gender                   29452 non-null object
age                      29452 non-null float64
engaged_last_30          29452 non-null object
lifecycle                29452 non-null object
days_since_last_order    29452 non-null float64
previous_order_amount    29452 non-null float64
3rd_party_stores         29452 non-null int64
dtypes: float64(4), int64(1), object(3)
memory usage: 1.8+ MB
```

In [11]:
```
red=pd.get_dummies(red)#re-coding object variables
```

In [12]:
```
red.head()
```

Out[12]:

arty_stores	gender_0.0	gender_1.0	gender_unknown	engaged_last_30_0.0	engaged_last_30_1.0	engaged_last_30_unknown	lifecycle_A	lifecycle_B	lifecycle_C
0	0	1	0	1	0	0	0	1	0
0	0	1	0	1	0	0	1	0	0
1	0	1	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	1
4	0	0	1	0	0	1	0	0	1

In [13]:
```
#delete redundancy
del red['gender_unknown']
del red['engaged_last_30_unknown']
del red['lifecycle_C']
red.head()
```

Correlation analysis

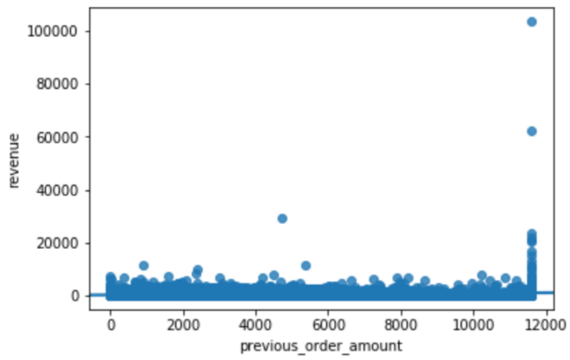
n [14]:
```
#check the correlation with revenue
red.corr()[['revenue']].sort_values('revenue',ascending=False)
#result shows the poor correlation between revenue and any one of the factors
```

Out[14]:

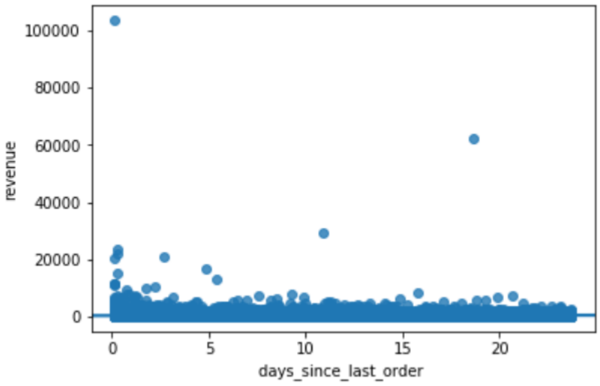
	revenue
revenue	1.000000
previous_order_amount	0.168540
engaged_last_30_1.0	0.038287
days_since_last_order	0.036654
lifecycle_A	0.013683
lifecycle_B	-0.008651
gender_1.0	-0.012422
gender_0.0	-0.014914
3rd_party_stores	-0.026398
engaged_last_30_0.0	-0.033274
age	-0.035801

In [15]:
```
sns.regplot('previous_order_amount','revenue',red) #select previous_order_amount plot the correlation chart
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0xbf81eb8>



In [95]:
```
sns.regplot('days_since_last_order ','revenue',red)
```
Out[95]: <matplotlib.axes._subplots.AxesSubplot at 0x1355ff98>



Data Modeling

In [19]:
```
#linear regression analysis
from sklearn.linear_model import LinearRegression
```

In [20]:
```
model=LinearRegression()
```

In [57]:
```
#select top 4 factors>0 for x
x=red[['previous_order_amount','engaged_last_30_1.0','days_since_last_order ','lifecycle_A']]
y=red['revenue'].values
```

In [22]:
```
import statsmodels.api as sm
```


In [22]:
```
import statsmodels.api as sm
```


In [58]:
```
#select the best group according to AIC value, the smaller the better
predictorcols=['previous_order_amount','engaged_last_30_1.0','days_since_last_order ','lifecycle_A']
import itertools
AICs={}
for k in range(1,len(predictorcols)+1):
    for variables in itertools.combinations(predictorcols,k):
        predictors=x[list(variables)]
        predictors2=sm.add_constant(predictors)
        est=sm.OLS(y,predictors2)
        res=est.fit()
        AICs[variables]=res.aic
```

In [59]:
```
#show AIC values rank group
from collections import Counter
c=Counter(AICs)
c.most_common()[::-1]
```

Out[59]:
```
[(('previous_order_amount',
   'engaged_last_30_1.0',
   'days_since_last_order ',
   'lifecycle_A'),
  487149.1109399847),
 (('previous_order_amount', 'days_since_last_order ', 'lifecycle_A'),
  487152.61894204706),
 (('previous_order_amount', 'engaged_last_30_1.0', 'days_since_last_order '),
  487159.6252754777),
 (('previous_order_amount', 'days_since_last_order '), 487163.10989113926),
 (('previous_order_amount', 'engaged_last_30_1.0'), 487237.58110378217),
 (('previous_order_amount', 'engaged_last_30_1.0', 'lifecycle_A'),
  487239.4107417161),
 (('previous_order_amount',), 487240.7190121286),
 (('previous_order_amount', 'lifecycle_A'), 487242.55263833655),
 (('engaged_last_30_1.0', 'days_since_last_order ', 'lifecycle_A'),
  487979.9462825058),
 (('engaged_last_30_1.0', 'days_since_last_order '), 488006.81572586216),
 (('days_since_last_order ', 'lifecycle_A'), 488022.3928823604),
 (('engaged_last_30_1.0', 'lifecycle_A'), 488043.20461027825),
 (('engaged_last_30_1.0',), 488046.23247214034),
 (('days_since_last_order ',), 488049.84043813415),
 (('lifecycle_A',), 488083.92272080784)]
```

In [73]:
```
#select the 1st group in AIC rank list,try linearregression first
model.fit(x,y)
```

Out[73]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [74]:
```
#Summary,check the R-square(larger is better),AIC(smaller is better)
x_=sm.add_constant(x)
est=sm.OLS(y,x_)
est2=est.fit()
print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.032
Model:                            OLS   Adj. R-squared:                  0.031
Method:                 Least Squares   F-statistic:                     240.4
Date:                Sat, 10 Aug 2019   Prob (F-statistic):          1.63e-203
Time:                        12:24:02   Log-Likelihood:            -2.4357e+05
No. Observations:               29452   AIC:                         4.871e+05
Df Residuals:                   29447   BIC:                         4.872e+05
Df Model:                           4
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                    157.7448     11.516     13.698      0.000     135.173     180.316
previous_order_amount      0.0684      0.002     29.062      0.000       0.064       0.073
engaged_last_30_1.0       63.7613     27.169      2.347      0.019      10.508     117.014
days_since_last_order      8.9654      0.933      9.614      0.000       7.138      10.793
lifecycle_A               65.8713     18.620      3.538      0.000      29.375     102.368
------------------------------------------------------------------------------
```

In [75]:
```
#comepare the RMSE&MAE
score=model.score(x,y)
predictions=model.predict(x)
error=predictions-y
rmse=(error**2).mean()**.5
mae=abs(error).mean()

print(rmse)
print(mae)
```

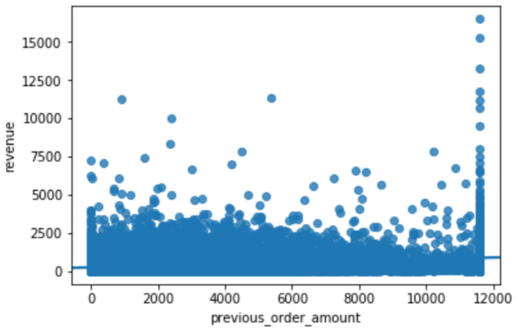
```
944.9319576208536
352.8277659947816
```

filter data again, model assessment and optimization

In [64]:
```
red1=red[red['revenue']<20000]#from the chart can see there's few points which revenue larger than 20000,filter
```

In [65]:
```
sns.regplot('previous_order_amount','revenue',red1)#plot again
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0xfe38898>



In [81]:
```
#select the 1st group from AIC rank list
X1=red1[['previous_order_amount','engaged_last_30_1.0','days_since_last_order ','lifecycle_A']]
Y1=red1['revenue'].values
```

In [101]:
```
model.fit(X1,Y1)
```

Out[101]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [83]:
```
X1_=sm.add_constant(X1)
est=sm.OLS(Y1,X1_)
est2=est.fit()
print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.056
Model:                            OLS   Adj. R-squared:                  0.056
Method:                 Least Squares   F-statistic:                     438.9
Date:                Sat, 10 Aug 2019   Prob (F-statistic):               0.00
Time:                        12:33:24   Log-Likelihood:            -2.2852e+05
No. Observations:               29445   AIC:                         4.570e+05
Df Residuals:                   29440   BIC:                         4.571e+05
Df Model:                           4
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
const                  186.9997      6.922     27.016      0.000     173.433     200.567
previous_order_amount    0.0543      0.001     38.337      0.000       0.052       0.057
engaged_last_30_1.0     75.5621     16.334      4.626      0.000      43.547     107.577
days_since_last_order    8.5395      0.560     15.237      0.000       7.441       9.638
lifecycle_A             42.9239     11.192      3.835      0.000      20.987      64.861
==============================================================================
```

In [84]:
```
score=model.score(X1,Y1)
predictions=model.predict(X1)
error=predictions-Y1
rmse=(error**2).mean()**.5
mae=abs(error).mean()

print(rmse)
print(mae)
```

```
567.8385956494542
339.6992433398352
```

Conclusion： Through the data visualization and linear regression analysis, found that there's little correlation between the revenue and each factor; after model assessment and optimization, selected the data filtered the points which not in the baseline for modeling:

Revenue=186.9997+0.0543*previous_order_amount+75.5621*engaged_last_30_1.0+8.5395*days_since_last_order+42.9239*lifecycle_A

R^2=0.056

AIC=4.570e+05

RMSE=567.84

MAE=339.70