Giving Customer Subscription Offer Based on Customer Behavior using Fintech Application

By Mabtoor Mabx [https://linkedin.com/in/mabtoor-mabx (https://linkedin.com/in/mabtoor-mabx)](https://linkedin.com/in/mabtoor-mabx)

Project Source [https://github.com/Mabtoor-Mabx/Machine-Learning-Projects (https://github.com/Mabtoor-Mabx/Machine-Learning-Projects)](https://github.com/Mabtoor-Mabx/Machine-Learning-Projects)

Goal Of Our Project

The FinTech Company is Launced Application for Both Android and IOS Users and They want to grow their business. They Have no idea how many people is ready to accept their premium offers so they decide to allow all users to enjoy premium offer free of cost for first 24 hours to check whether they used premium services or not and collect the information. After that company is ready to accept for those people who is not sure about buying the premium cost

According To Given Condition, This is Classification Problem

Import Libraries

In [1]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from dateutil import parser
```

Import Dataset and Explore Dataset

In [2]:
```
fintech_app_dataset = pd.read_csv('FineTech_appData.csv')
```

In [3]:
```
fintech_app_dataset.head()
```

Out[3]:

	user	first_open	dayofweek	hour	age	screen_list
0	235136	2012-12-27 02:14:51.273	3	02:00:00	23	idscreen,joinscreen,Cycle,product_review,ScanP...
1	333588	2012-12-02 01:16:00.905	6	01:00:00	24	joinscreen,product_review,product_review2,Scan...
2	254414	2013-03-19 19:19:09.157	1	19:00:00	23	Splash,Cycle,Loan
3	234192	2013-07-05 16:08:46.354	4	16:00:00	28	product_review,Home,product_review,Loan3,Finan...
4	51549	2013-02-26 18:50:48.661	1	18:00:00	31	idscreen,joinscreen,Cycle,Credit3Container,Sca...

In [4]:
```
fintech_app_dataset.shape
```

Out[4]:
```
(50000, 12)
```

In [5]:
```
fintech_app_dataset.tail(6)
```

Out[5]:

	user	first_open	dayofweek	hour	age	screen_
49994	90813	2013-02-25 19:35:12.691	0	19:00:00	36	idscreen,joinscreen,Cycle,product_review,proc
49995	222774	2013-05-09 13:46:17.871	3	13:00:00	32	Splash,Home,ScanPreview,VerifyPhone,VerifySSI
49996	169179	2013-04-09 00:05:17.823	1	00:00:00	35	Cycle,Splash,Home,RewardsConta
49997	302367	2013-02-20 22:41:51.165	2	22:00:00	39	joinscreen,product_review,product_review2,Sca
49998	324905	2013-04-28 12:33:04.288	6	12:00:00	27	Cycle,Home,product_review,product_review,proc
49999	27047	2012-12-14 01:22:44.638	4	01:00:00	25	product_review,ScanPreview,VerifyDateOfBirth

In [6]:
```
for i in [1,2,3,4,5]:
    print(fintech_app_dataset.loc[i,'screen_list'], '\n')
```

```
joinscreen,product_review,product_review2,ScanPreview,VerifyDateOfBirth,locatio
n,VerifyCountry,VerifyPhone,VerifyToken,Institutions,Loan2

Splash,Cycle,Loan

product_review,Home,product_review,Loan3,Finances,Credit3,ReferralContainer,Lea
derboard,Rewards,RewardDetail,ScanPreview,location,VerifyDateOfBirth,VerifyPhon
e,VerifySSN,Credit1,Credit2

idscreen,joinscreen,Cycle,Credit3Container,ScanPreview,VerifyPhone,VerifySSN,Cr
edit1,Loan2,Home,Institutions,SelectInstitution,BankVerification,ReferralContai
ner,product_review,product_review2,VerifyCountry,VerifyToken,product_review

idscreen,Cycle,Home,ScanPreview,VerifyPhone,VerifySSN,Credit1,Credit3Dashboard,
Loan2,Institutions,product_review,product_review,product_review3
```

Checking Null Values

In [7]:
```
fintech_app_dataset.isnull().sum()
```

Out[7]:
```
user                      0
first_open                0
dayofweek                 0
hour                      0
age                       0
screen_list               0
numscreens                0
minigame                  0
used_premium_feature      0
enrolled                  0
enrolled_date         18926
liked                     0
dtype: int64
```

As we can see that Only Enrolled_date have Missing Value

In [8]:
```
fintech_app_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   user                  50000 non-null  int64 
 1   first_open            50000 non-null  object
 2   dayofweek             50000 non-null  int64 
 3   hour                  50000 non-null  object
 4   age                   50000 non-null  int64 
 5   screen_list           50000 non-null  object
 6   numscreens            50000 non-null  int64 
 7   minigame              50000 non-null  int64 
 8   used_premium_feature  50000 non-null  int64 
 9   enrolled              50000 non-null  int64 
 10  enrolled_date         31074 non-null  object
 11  liked                 50000 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 4.6+ MB
```

In [9]:
```
fintech_app_dataset.describe()
```

Out[9]:

	user	dayofweek	age	numscreens	minigame	used_premium_featu
count	50000.000000	50000.000000	50000.00000	50000.000000	50000.000000	50000.00000
mean	186889.729900	3.029860	31.72436	21.095900	0.107820	0.17202
std	107768.520361	2.031997	10.80331	15.728812	0.310156	0.37740
min	13.000000	0.000000	16.00000	1.000000	0.000000	0.00000
25%	93526.750000	1.000000	24.00000	10.000000	0.000000	0.00000
50%	187193.500000	3.000000	29.00000	18.000000	0.000000	0.00000
75%	279984.250000	5.000000	37.00000	28.000000	0.000000	0.00000
max	373662.000000	6.000000	101.00000	325.000000	1.000000	1.00000

Get Unique Value of Each Columns and Columns Length

In [10]:
```
features = fintech_app_dataset.columns
for i in features:
    print('''Unique Value of {}\n{}\n len is {} \n.........................\r
    '''.format(i, fintech_app_dataset[i].unique(), len(fintech_app_dataset[i].uni
```

```
Unique Value of user
[235136 333588 254414 ... 302367 324905  27047]
 len is 49874
.........................


Unique Value of first_open
['2012-12-27 02:14:51.273' '2012-12-02 01:16:00.905'
 '2013-03-19 19:19:09.157' ... '2013-02-20 22:41:51.165'
 '2013-04-28 12:33:04.288' '2012-12-14 01:22:44.638']
 len is 49747
.........................


Unique Value of dayofweek
[3 6 1 4 2 0 5]
 len is 7
.........................


Unique Value of hour
[' 02:00:00' ' 01:00:00' ' 19:00:00' ' 16:00:00' ' 18:00:00' ' 09:00:00'
 ' 03:00:00' ' 14:00:00' ' 04:00:00' ' 11:00:00' ' 06:00:00' ' 21:00:00'
 ' 05:00:00' ' 17:00:00' ' 20:00:00' ' 00:00:00' ' 22:00:00' ' 10:00:00'
 ' 08:00:00' ' 15:00:00' ' 13:00:00' ' 23:00:00' ' 12:00:00' ' 07:00:00']
 len is 24
.........................


Unique Value of age
[ 23  24  28  31  20  35  26  29  39  32  25  17  21  55  38  27  48  37
  22  36  30  58  40  33  57  19  45  34  46  56  42  43  41  47  18  53
  44  49  60  50  52  62  63  16  54  70  51  69  68  59  76  75  66  61
  72  65  90  64  67  73  77  71  74  89  78  86  80  82  79  87  81  85
 101  88  83 100  84  98]
 len is 78
.........................


Unique Value of screen_list
['idscreen,joinscreen,Cycle,product_review,ScanPreview,VerifyDateOfBirth,Verify
Phone,VerifyToken,ProfileVerifySSN,Loan2,Settings,ForgotPassword,Login'
 'joinscreen,product_review,product_review2,ScanPreview,VerifyDateOfBirth,locat
ion,VerifyCountry,VerifyPhone,VerifyToken,Institutions,Loan2'
 'Splash,Cycle,Loan' ...
 'joinscreen,product_review,product_review2,ScanPreview,VerifyCountry,VerifyPho
ne,VerifyToken,VerifyDateOfBirth,location,Home'
 'Cycle,Home,product_review,product_review,product_review3,ScanPreview,VerifyDa
teOfBirth,location,VerifyCountry,VerifyPhone,VerifyToken,product_review,product
_review,VerifySSN,product_review,SelectInstitution,BankVerification,product_rev
iew,product_review'
 'product_review,ScanPreview,VerifyDateOfBirth,VerifyCountry,ProfileVerifySSN,P
```

```
rofilePage,ProfileEducation,ProfileEducationMajor,Saving2Amount,Saving8,Profile
MaritalStatus,ProfileChildren,Saving2,Saving9,Saving7,Saving6,Saving5,Home,Loan
2']
 len is 38799
.............................


Unique Value of numscreens
[ 15  13   3  40  32  14  41  33  19  25  11   4   9  26   6  20   5   8
  42   1  38  49  35  10  52  50  76  37  16  47  90  24  45  31  39  17
  28  27  57  23  21  12   7  18  48  29 136  34  59  89  22  43  36  56
  30   2  44  92  51  70  58  66  46  55  61  75  71  78  85  62  53  54
  73  68  69  63  64  88 106  80 127  74  72 137  83  77  65 104  60  67
  94  81 110  91  82  96 165  79  86 116  99  98 187  84 111 109 107 162
  97 100  95  87 122 216 115 102 128 234 112 108 114 125 119  93 185 192
 189 153 243 103 101 118 325 141 129 133 126 120 123 134 121 105 113 117
 200 247 179 132 144 130 148]
 len is 151
.............................


Unique Value of minigame
[0 1]
 len is 2
.............................


Unique Value of used_premium_feature
[0 1]
 len is 2
.............................


Unique Value of enrolled
[0 1]
 len is 2
.............................


Unique Value of enrolled_date
[nan '2013-07-05 16:11:49.513' '2013-02-26 18:56:37.841' ...
 '2013-02-25 19:36:56.082' '2013-05-09 13:47:52.875'
 '2013-04-28 12:35:38.709']
 len is 31002
.............................


Unique Value of liked
[0 1]
 len is 2
.............................
```

Checking Dtypes of Columns

In [11]:
```
fintech_app_dataset.dtypes
```

Out[11]:
```
user                    int64
first_open             object
dayofweek               int64
hour                   object
age                     int64
screen_list            object
numscreens              int64
minigame                int64
used_premium_feature    int64
enrolled                int64
enrolled_date          object
liked                   int64
dtype: object
```

In [12]:
```
fintech_app_dataset.columns
```

Out[12]:
```
Index(['user', 'first_open', 'dayofweek', 'hour', 'age', 'screen_list',
       'numscreens', 'minigame', 'used_premium_feature', 'enrolled',
       'enrolled_date', 'liked'],
      dtype='object')
```

Dropping Object Datatype of Columns

In [13]:
```
fintech_app_dataset_2 = fintech_app_dataset.drop(['user', 'first_open','screen_li
```

In [14]:
```
fintech_app_dataset_2.head(6)
```

Out[14]:

	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	liked
0	3	02:00:00	23	15	0	0	0	0
1	6	01:00:00	24	13	0	0	0	0
2	1	19:00:00	23	3	0	1	0	1
3	4	16:00:00	28	40	0	0	1	0
4	1	18:00:00	31	32	0	0	1	1
5	2	09:00:00	20	14	0	0	1	0

Hour Data Convert it into string

In [15]:
```
fintech_app_dataset_2['hour']= fintech_app_dataset_2.hour.str.slice(1,3).astype(i
```

In [16]:
```
fintech_app_dataset['hour']= fintech_app_dataset.hour.str.slice(1,3).astype(int)
```

In [17]:
```
fintech_app_dataset_2.head(6)
```

Out[17]:

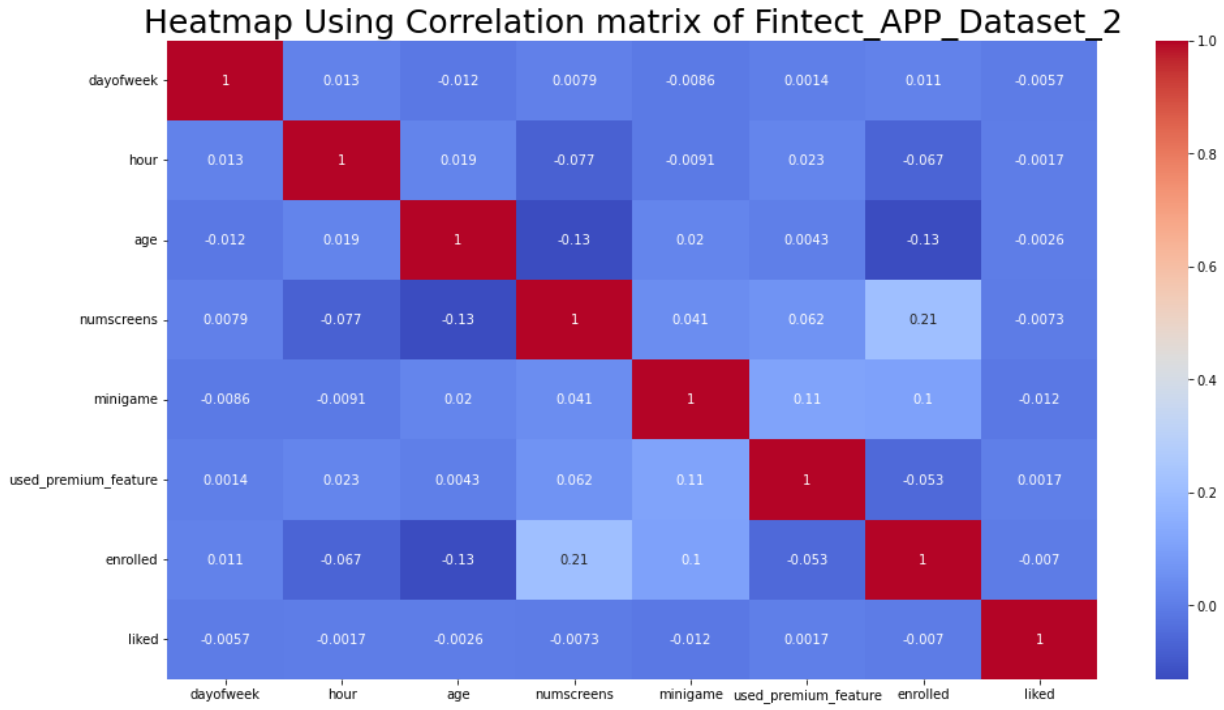
	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	liked
0	3	2	23	15	0	0	0	0
1	6	1	24	13	0	0	0	0
2	1	19	23	3	0	1	0	1
3	4	16	28	40	0	0	1	0
4	1	18	31	32	0	0	1	1
5	2	9	20	14	0	0	1	0

Data Visualization

Heatmap Using Correlation matrix

In [18]:
```
# Heatmap

plt.figure(figsize=(16,9))

sns.heatmap(fintech_app_dataset_2.corr(), annot=True, cmap='coolwarm')

plt.title('Heatmap Using Correlation matrix of Fintect_APP_Dataset_2', fontsize=2
```

Out[18]: Text(0.5, 1.0, 'Heatmap Using Correlation matrix of Fintect_APP_Dataset_2')



Pairplot of Fintect_App_Dataset_2

In [19]:
```
# Pairplot of Fintect_APP_dataset_2

sns.pairplot(fintech_app_dataset_2, hue='enrolled')
```

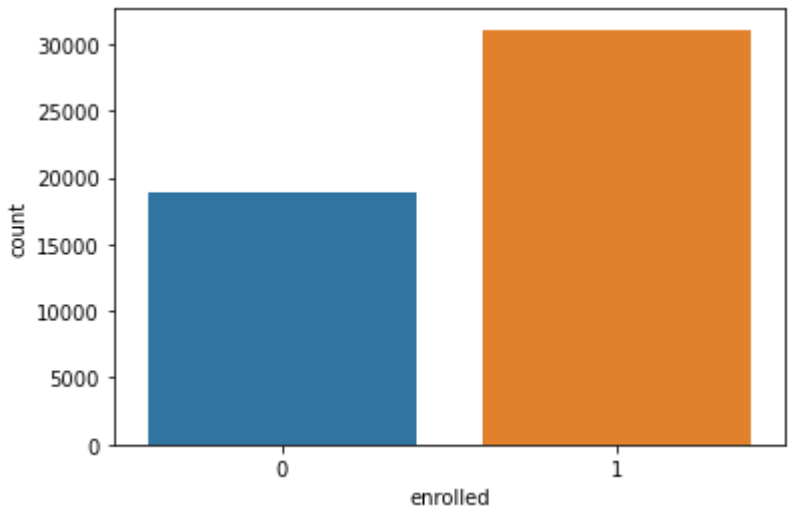
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x19a492ad490>
```

Counterplot of Enrolled Feature

In [20]:
```
# Show Counterplot of Enrolled Feature

sns.countplot(fintech_app_dataset_2.enrolled)
```

```
D:\Annaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass t
he following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[20]: <AxesSubplot:xlabel='enrolled', ylabel='count'>



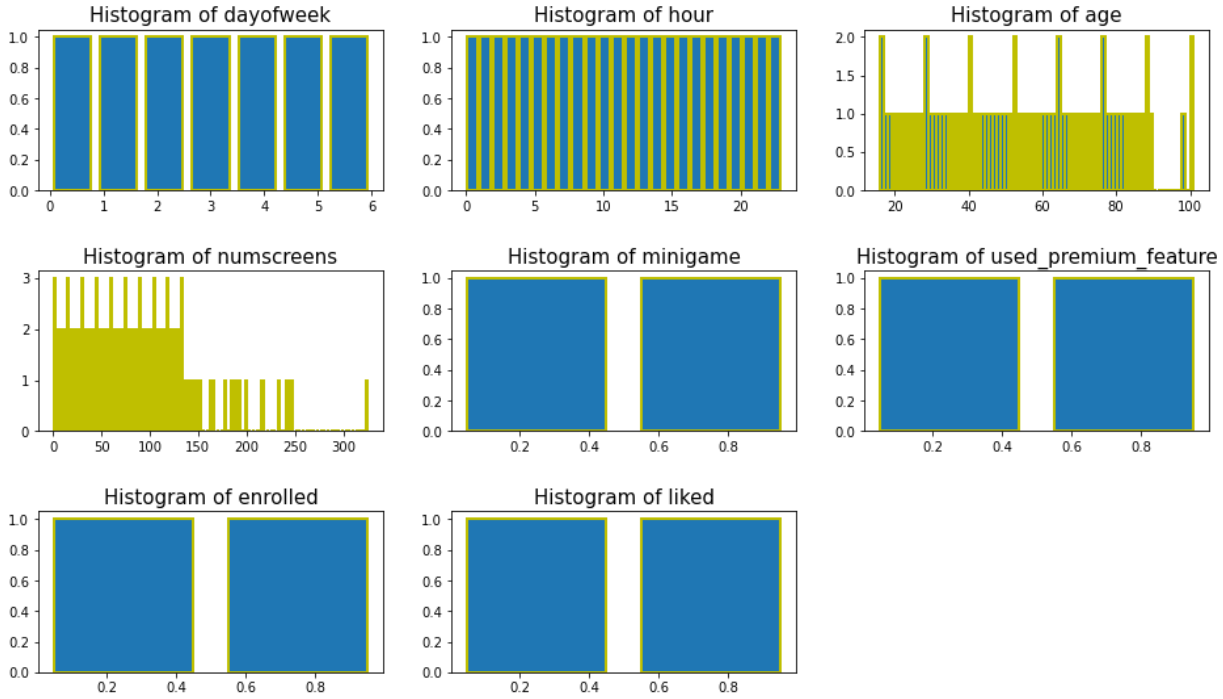
Checking Enrolled or Not Enrolled Users

In [21]:
```
print('Not Enrolled Users = ', (fintech_app_dataset_2.enrolled < 1).sum(), 'Out o
print('Enrolled Users =  ', (fintech_app_dataset_2.enrolled >= 1).sum() , 'Out of
```

```
Not Enrolled Users =  18926 Out of 50000
Enrolled Users =   31074 Out of 50000
```

Histogram of Each Feature of Fintech_APP_dataset_2

In [22]:
```
# Ploy Histogram

plt.figure(figsize=(16,9))
features = fintech_app_dataset_2.columns
for i , j in enumerate(features):
    plt.subplot(3,3,i+1)
    plt.title('Histogram of {}'.format(j), fontsize=15)

    bins = len(fintech_app_dataset_2[j].unique())
    plt.hist(fintech_app_dataset_2[j].unique(), bins=bins, rwidth=0.8, edgecolor=
plt.subplots_adjust(hspace=0.5)
```



Features list

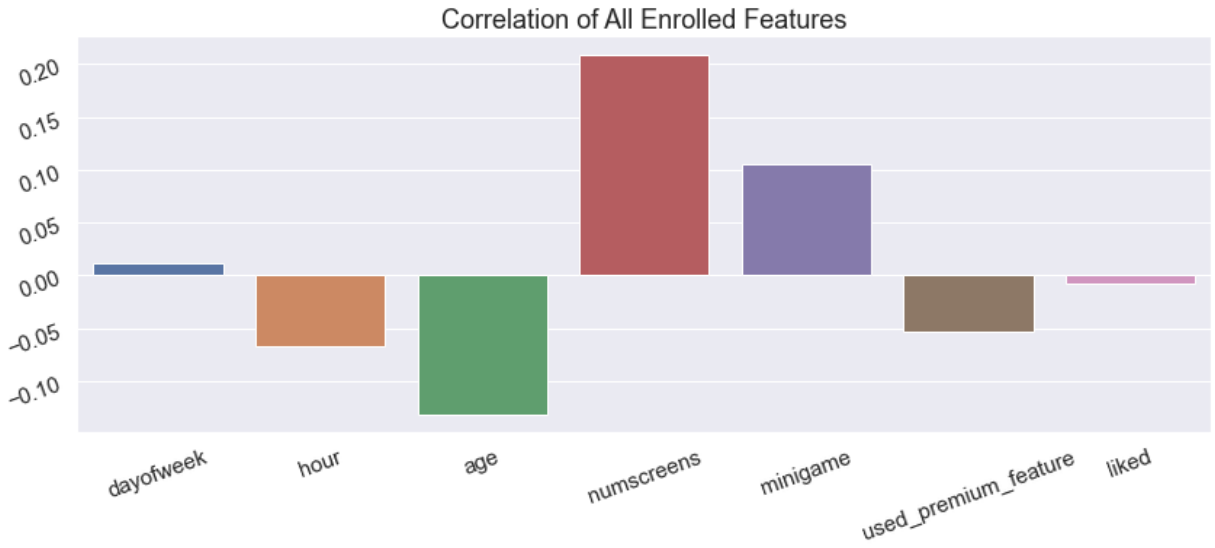
In [23]:
```
for i,j in enumerate(features):
    print(i,j)
```

```
0 dayofweek
1 hour
2 age
3 numscreens
4 minigame
5 used_premium_feature
6 enrolled
7 liked
```

Correlation Barplot with Enrolled Feature

In [24]:
```
# Show Correlation Barplot

sns.set()
plt.figure(figsize=(14,5))
plt.title('Correlation of All Enrolled Features', fontsize=18)
fintech_app_dataset_3 = fintech_app_dataset_2.drop(['enrolled'], axis=1)
ax = sns.barplot(fintech_app_dataset_3.columns, fintech_app_dataset_3.corrwith(fi
ax.tick_params(labelsize=15, labelrotation=20, color='k')
```

```
D:\Annaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Parsing Object Data in Date and Time Format

In [25]:
```
fintech_app_dataset['first_open'] = [parser.parse(i) for i in fintech_app_dataset
```

In [26]:
```
fintech_app_dataset['enrolled_date'] = [parser.parse(i) if isinstance(i,str) else
```

In [27]:
```
fintech_app_dataset.dtypes
```

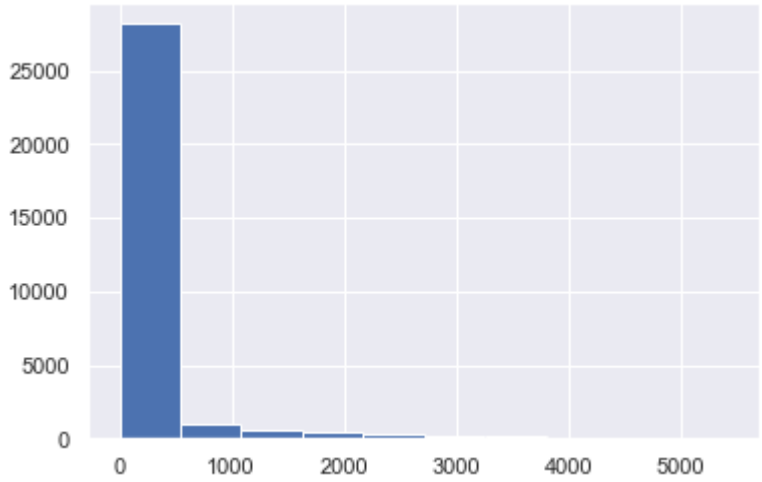
Out[27]:
```
user                            int64
first_open              datetime64[ns]
dayofweek                       int64
hour                            int32
age                             int64
screen_list                    object
numscreens                      int64
minigame                        int64
used_premium_feature            int64
enrolled                        int64
enrolled_date           datetime64[ns]
liked                           int64
dtype: object
```

In [28]:
```
fintech_app_dataset['time_to_enrolled'] = (fintech_app_dataset.enrolled_date - fi
```

Plot Histogram

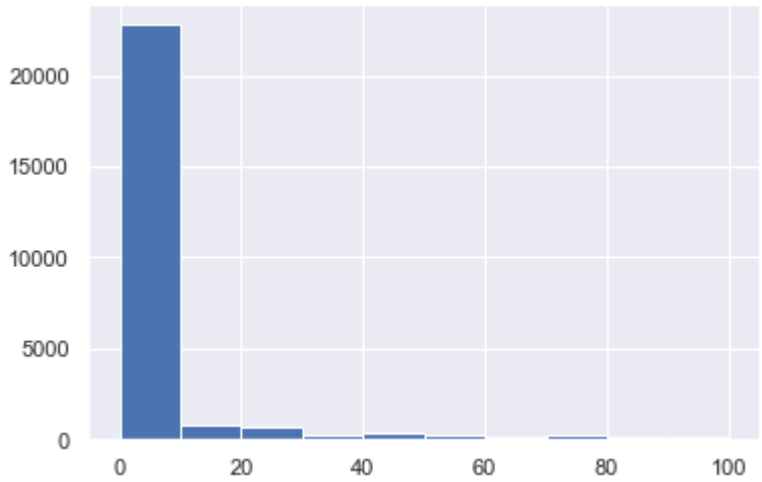
In [29]:
```
plt.hist(fintech_app_dataset['time_to_enrolled'].dropna())
```

Out[29]:
```
(array([2.8195e+04, 1.0320e+03, 5.6700e+02, 4.2500e+02, 2.8800e+02,
        1.7900e+02, 1.6500e+02, 9.7000e+01, 1.0400e+02, 2.2000e+01]),
 array([   0. ,  543.4, 1086.8, 1630.2, 2173.6, 2717. , 3260.4, 3803.8,
        4347.2, 4890.6, 5434. ]),
 <BarContainer object of 10 artists>)
```



In [30]:
```
plt.hist(fintech_app_dataset['time_to_enrolled'].dropna(), range=(0,100))
```

Out[30]:
```
(array([22793.,   755.,   707.,   288.,   347.,   210.,   187.,   212.,
          135.,   194.]),
 array([  0.,  10.,  20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 100.]),
 <BarContainer object of 10 artists>)
```



Customers Who Enrolled After 48 Hours Consider as 0

In [31]:
```
fintech_app_dataset.loc[fintech_app_dataset.time_to_enrolled > 48, 'enrolled'] =0
```

In [32]:
```
fintech_app_dataset
```

Out[32]:

	user	first_open	dayofweek	hour	age	screen_list
0	235136	2012-12-27 02:14:51.273	3	2	23	idscreen,joinscreen,Cycle,product_review,ScanP...
1	333588	2012-12-02 01:16:00.905	6	1	24	joinscreen,product_review,product_review2,Scan...
2	254414	2013-03-19 19:19:09.157	1	19	23	Splash,Cycle,Loan
3	234192	2013-07-05 16:08:46.354	4	16	28	product_review,Home,product_review,Loan3,Finan...
4	51549	2013-02-26 18:50:48.661	1	18	31	idscreen,joinscreen,Cycle,Credit3Container,Sca...
...	...	...	...	...	...	...
49995	222774	2013-05-09 13:46:17.871	3	13	32	Splash,Home,ScanPreview,VerifyPhone,VerifySSN,...
49996	169179	2013-04-09 00:05:17.823	1	0	35	Cycle,Splash,Home,RewardsContainer
49997	302367	2013-02-20 22:41:51.165	2	22	39	joinscreen,product_review,product_review2,Scan...
49998	324905	2013-04-28 12:33:04.288	6	12	27	Cycle,Home,product_review,product_review,produ...
49999	27047	2012-12-14 01:22:44.638	4	1	25	product_review,ScanPreview,VerifyDateOfBirth,V...

50000 rows × 13 columns

In [33]:
```
fintech_app_dataset.drop(columns=['time_to_enrolled', 'enrolled_date', 'first_ope
```

In [34]:
```
fintech_app_dataset
```

Out[34]:

	user	dayofweek	hour	age	screen_list	numscreens
0	235136	3	2	23	idscreen,joinscreen,Cycle,product_review,ScanP...	15
1	333588	6	1	24	joinscreen,product_review,product_review2,Scan...	13
2	254414	1	19	23	Splash,Cycle,Loan	3
3	234192	4	16	28	product_review,Home,product_review,Loan3,Finan...	40
4	51549	1	18	31	idscreen,joinscreen,Cycle,Credit3Container,Sca...	32
...	...	...	...	...	...	...
49995	222774	3	13	32	Splash,Home,ScanPreview,VerifyPhone,VerifySSN,...	13
49996	169179	1	0	35	Cycle,Splash,Home,RewardsContainer	4
49997	302367	2	22	39	joinscreen,product_review,product_review2,Scan...	25
49998	324905	6	12	27	Cycle,Home,product_review,product_review,produ...	26
49999	27047	4	1	25	product_review,ScanPreview,VerifyDateOfBirth,V...	26

50000 rows × 10 columns

Convert CSV File Into Numpy Format

In [35]:
```
fintech_app_screen_dataset = pd.read_csv('top_screens.csv').top_screens.values
```

In [36]:
```
fintech_app_screen_dataset
```

Out[36]:
```
array(['Loan2', 'location', 'Institutions', 'Credit3Container',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth',
       'ProfilePage', 'VerifyCountry', 'Cycle', 'idscreen',
       'Credit3Dashboard', 'Loan3', 'CC1Category', 'Splash', 'Loan',
       'CC1', 'RewardsContainer', 'Credit3', 'Credit1', 'EditProfile',
       'Credit2', 'Finances', 'CC3', 'Saving9', 'Saving1', 'Alerts',
       'Saving8', 'Saving10', 'Leaderboard', 'Saving4', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'Saving7', 'ProfileEducationMajor', 'Rewards', 'AccountView',
       'VerifyAnnualIncome', 'VerifyIncomeType', 'Saving2', 'Saving6',
       'Saving2Amount', 'Saving5', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'Loan4',
       'ResendToken', 'TransactionList', 'NetworkFailure', 'ListPicker'],
      dtype=object)
```

In [37]:
```
type(fintech_app_screen_dataset)
```

Out[37]:
```
numpy.ndarray
```

Add , at the End of Screen list for Further Operations

In [38]:
```
fintech_app_dataset['screen_list'] = fintech_app_dataset.screen_list.astype(str)
```

In [39]:
```
fintech_app_dataset
```

Out[39]:

	user	dayofweek	hour	age	screen_list	numscreens
0	235136	3	2	23	idscreen,joinscreen,Cycle,product_review,ScanP...	15
1	333588	6	1	24	joinscreen,product_review,product_review2,Scan...	13
2	254414	1	19	23	Splash,Cycle,Loan,	3
3	234192	4	16	28	product_review,Home,product_review,Loan3,Finan...	40
4	51549	1	18	31	idscreen,joinscreen,Cycle,Credit3Container,Sca...	32
...	...	...	...	...	...	...
49995	222774	3	13	32	Splash,Home,ScanPreview,VerifyPhone,VerifySSN,...	13
49996	169179	1	0	35	Cycle,Splash,Home,RewardsContainer,	4
49997	302367	2	22	39	joinscreen,product_review,product_review2,Scan...	25
49998	324905	6	12	27	Cycle,Home,product_review,product_review,produ...	26
49999	27047	4	1	25	product_review,ScanPreview,VerifyDateOfBirth,V...	26

50000 rows × 10 columns

Convert String into Number

In [40]:
```
for screen_name in fintech_app_screen_dataset:
    fintech_app_dataset[screen_name]= fintech_app_dataset.screen_list.str.contair
    fintech_app_dataset['screen_list'] = fintech_app_dataset.screen_list.str.repl
```

In [41]:
```
# Test

fintech_app_dataset.screen_list.str.contains('Splash').astype(int)
```

Out[41]:
```
0        0
1        0
2        0
3        0
4        0
        ..
49995    0
49996    0
49997    0
49998    0
49999    0
Name: screen_list, Length: 50000, dtype: int32
```

In [42]:
```
fintech_app_dataset.screen_list.str.replace('Splash'+',', '')
```

Out[42]:
```
0        joinscreen,product_review,ScanPreview,VerifyTo...
1        joinscreen,product_review,product_review2,Scan...
2
3        product_review,Home,product_review,ReferralCon...
4        joinscreen,ScanPreview,VerifySSN,Home,SelectIn...
                              ...
49995    Home,ScanPreview,VerifySSN,product_review,prod...
49996                                                Home,
49997    joinscreen,product_review,product_review2,Scan...
49998    Home,product_review,product_review,product_rev...
49999    product_review,ScanPreview,ProfileVerifySSN,Pr...
Name: screen_list, Length: 50000, dtype: object
```

In [43]:
```
# Get Shape

fintech_app_dataset.shape
```

Out[43]:
```
(50000, 68)
```

In [44]:
```
# Head of DataFrame

fintech_app_dataset.head(8)
```

Out[44]:

	user	dayofweek	hour	age	screen_list	numscreens	mi
0	235136	3	2	23	joinscreen,product_review,ScanPreview,VerifyTo...	15	
1	333588	6	1	24	joinscreen,product_review,product_review2,Scan...	13	
2	254414	1	19	23		3	
3	234192	4	16	28	product_review,Home,product_review,ReferralCon...	40	
4	51549	1	18	31	joinscreen,ScanPreview,VerifySSN,Home,SelectIn...	32	
5	56480	2	9	20	Home,ScanPreview,VerifySSN,product_review,prod...	14	
6	144649	1	2	35	product_review,product_review2,ScanPreview,	3	
7	249366	1	3	26	Home,product_review,product_review2,ScanPrevie...	41	

8 rows × 68 columns

In [45]:
```
# Remain Screen in ScreenList

fintech_app_dataset.loc[0, 'screen_list']
```

Out[45]:
```
'joinscreen,product_review,ScanPreview,VerifyToken,ProfileVerifySSN,Settings,Fo
rgotPassword,'
```

In [46]:
```
fintech_app_dataset.screen_list.str.count(',').head(7)
```

Out[46]:
```
0     7
1     5
2     0
3     6
4    10
5     6
6     3
Name: screen_list, dtype: int64
```

Count Remain Screen List and Store it in Remain_Screen_List

In [47]:
```
fintech_app_dataset['remain_screen_list'] = fintech_app_dataset.screen_list.str.c
```

In [48]:
```
# Drop Screen List

fintech_app_dataset.drop(columns= ['screen_list'], inplace=True)
```

In [49]:
```
fintech_app_dataset
```

Out[49]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	lik
0	235136	3	2	23	15	0	0	0	
1	333588	6	1	24	13	0	0	0	
2	254414	1	19	23	3	0	1	0	
3	234192	4	16	28	40	0	0	1	
4	51549	1	18	31	32	0	0	1	
...	...	...	...	...	...	...	...	...	
49995	222774	3	13	32	13	0	0	1	
49996	169179	1	0	35	4	0	1	0	
49997	302367	2	22	39	25	0	0	0	
49998	324905	6	12	27	26	0	0	1	
49999	27047	4	1	25	26	0	0	0	

50000 rows × 68 columns

In [50]:
```
# Total Columns

fintech_app_dataset.columns
```

Out[50]:
```
Index(['user', 'dayofweek', 'hour', 'age', 'numscreens', 'minigame',
       'used_premium_feature', 'enrolled', 'liked', 'Loan2', 'location',
       'Institutions', 'Credit3Container', 'VerifyPhone', 'BankVerification',
       'VerifyDateOfBirth', 'ProfilePage', 'VerifyCountry', 'Cycle',
       'idscreen', 'Credit3Dashboard', 'Loan3', 'CC1Category', 'Splash',
       'Loan', 'CC1', 'RewardsContainer', 'Credit3', 'Credit1', 'EditProfile',
       'Credit2', 'Finances', 'CC3', 'Saving9', 'Saving1', 'Alerts', 'Saving8',
       'Saving10', 'Leaderboard', 'Saving4', 'VerifyMobile', 'VerifyHousing',
       'RewardDetail', 'VerifyHousingAmount', 'ProfileMaritalStatus',
       'ProfileChildren ', 'ProfileEducation', 'Saving7',
       'ProfileEducationMajor', 'Rewards', 'AccountView', 'VerifyAnnualIncome',
       'VerifyIncomeType', 'Saving2', 'Saving6', 'Saving2Amount', 'Saving5',
       'ProfileJobTitle', 'Login', 'ProfileEmploymentLength', 'WebView',
       'SecurityModal', 'Loan4', 'ResendToken', 'TransactionList',
       'NetworkFailure', 'ListPicker', 'remain_screen_list'],
      dtype='object')
```

Take Sum Of All Saving Screen in One Place

In [51]:
```
saving_screens = ['Saving1',
                  'Saving2',
                  'Saving2Amount',
                  'Saving4',
                  'Saving5',
                  'Saving6',
                  'Saving7',
                  'Saving8',
                  'Saving9',
                  'Saving10']

fintech_app_dataset['saving_screen_count'] = fintech_app_dataset[saving_screens].
fintech_app_dataset.drop(columns=saving_screens, inplace=True)
```

In [52]:
```
fintech_app_dataset
```

Out[52]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	lik
0	235136	3	2	23	15	0	0	0	
1	333588	6	1	24	13	0	0	0	
2	254414	1	19	23	3	0	1	0	
3	234192	4	16	28	40	0	0	1	
4	51549	1	18	31	32	0	0	1	
...	...	...	...	...	...	...	...	...	
49995	222774	3	13	32	13	0	0	1	
49996	169179	1	0	35	4	0	1	0	
49997	302367	2	22	39	25	0	0	0	
49998	324905	6	12	27	26	0	0	1	
49999	27047	4	1	25	26	0	0	0	

50000 rows × 59 columns

Count Credit Screens

In [53]:
```
credit_screens = ['Credit1',
                  'Credit2',
                  'Credit3',
                  'Credit3Container',
                  'Credit3Dashboard',]

fintech_app_dataset['credit_screen_counts'] = fintech_app_dataset[credit_screens]
fintech_app_dataset.drop(columns = credit_screens, inplace=True)
```

In [54]:
```
fintech_app_dataset
```

Out[54]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	lik
0	235136	3	2	23	15	0	0	0	
1	333588	6	1	24	13	0	0	0	
2	254414	1	19	23	3	0	1	0	
3	234192	4	16	28	40	0	0	1	
4	51549	1	18	31	32	0	0	1	
...	...	...	...	...	...	...	...	...	
49995	222774	3	13	32	13	0	0	1	
49996	169179	1	0	35	4	0	1	0	
49997	302367	2	22	39	25	0	0	0	
49998	324905	6	12	27	26	0	0	1	
49999	27047	4	1	25	26	0	0	0	

50000 rows × 55 columns

CC Screens

In [55]:
```
CC_screens = ['CC1',
              'CC1Category',
              'CC3']

fintech_app_dataset['screen_counts'] = fintech_app_dataset[CC_screens].sum(axis=1
fintech_app_dataset.drop(columns= CC_screens, inplace=True)
```

In [56]:
```
fintech_app_dataset
```

Out[56]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	lik
0	235136	3	2	23	15	0	0	0	
1	333588	6	1	24	13	0	0	0	
2	254414	1	19	23	3	0	1	0	
3	234192	4	16	28	40	0	0	1	
4	51549	1	18	31	32	0	0	1	
...	...	...	...	...	...	...	...	...	
49995	222774	3	13	32	13	0	0	1	
49996	169179	1	0	35	4	0	1	0	
49997	302367	2	22	39	25	0	0	0	
49998	324905	6	12	27	26	0	0	1	
49999	27047	4	1	25	26	0	0	0	

50000 rows × 53 columns

In [57]:
```
loan_screens = ['Loan',
                'Loan2',
                'Loan3',
                'Loan4']

fintech_app_dataset['loan_screens_counts'] =  fintech_app_dataset[loan_screens].s
fintech_app_dataset.drop(columns=loan_screens, inplace=True)
```

In [58]:
```
fintech_app_dataset
```

Out[58]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled
0	235136	3	2	23	15	0	0	0
1	333588	6	1	24	13	0	0	0
2	254414	1	19	23	3	0	1	0
3	234192	4	16	28	40	0	0	1
4	51549	1	18	31	32	0	0	1
...	...	...	...	...	...	...	...	...
49995	222774	3	13	32	13	0	0	1
49996	169179	1	0	35	4	0	1	0
49997	302367	2	22	39	25	0	0	0
49998	324905	6	12	27	26	0	0	1
49999	27047	4	1	25	26	0	0	0

50000 rows × 50 columns

In [59]:
```
fintech_app_dataset.shape
```

Out[59]:
```
(50000, 50)
```

In [60]:
```
fintech_app_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 50 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   user                      50000 non-null  int64
 1   dayofweek                 50000 non-null  int64
 2   hour                      50000 non-null  int32
 3   age                       50000 non-null  int64
 4   numscreens                50000 non-null  int64
 5   minigame                  50000 non-null  int64
 6   used_premium_feature      50000 non-null  int64
 7   enrolled                  50000 non-null  int64
 8   liked                     50000 non-null  int64
 9   location                  50000 non-null  int32
 10  Institutions              50000 non-null  int32
 11  VerifyPhone               50000 non-null  int32
 12  BankVerification          50000 non-null  int32
 13  VerifyDateOfBirth         50000 non-null  int32
 14  ProfilePage               50000 non-null  int32
 15  VerifyCountry             50000 non-null  int32
 16  Cycle                     50000 non-null  int32
 17  idscreen                  50000 non-null  int32
 18  Splash                    50000 non-null  int32
 19  RewardsContainer          50000 non-null  int32
 20  EditProfile               50000 non-null  int32
 21  Finances                  50000 non-null  int32
 22  Alerts                    50000 non-null  int32
 23  Leaderboard               50000 non-null  int32
 24  VerifyMobile              50000 non-null  int32
 25  VerifyHousing             50000 non-null  int32
 26  RewardDetail              50000 non-null  int32
 27  VerifyHousingAmount       50000 non-null  int32
 28  ProfileMaritalStatus      50000 non-null  int32
 29  ProfileChildren           50000 non-null  int32
 30  ProfileEducation          50000 non-null  int32
 31  ProfileEducationMajor     50000 non-null  int32
 32  Rewards                   50000 non-null  int32
 33  AccountView               50000 non-null  int32
 34  VerifyAnnualIncome        50000 non-null  int32
 35  VerifyIncomeType          50000 non-null  int32
 36  ProfileJobTitle           50000 non-null  int32
 37  Login                     50000 non-null  int32
 38  ProfileEmploymentLength   50000 non-null  int32
 39  WebView                   50000 non-null  int32
 40  SecurityModal             50000 non-null  int32
 41  ResendToken               50000 non-null  int32
 42  TransactionList           50000 non-null  int32
 43  NetworkFailure            50000 non-null  int32
 44  ListPicker                50000 non-null  int32
 45  remain_screen_list        50000 non-null  int64
 46  saving_screen_count       50000 non-null  int64
 47  credit_screen_counts      50000 non-null  int64
 48  screen_counts             50000 non-null  int64
 49  loan_screens_counts       50000 non-null  int64
```

```
dtypes: int32(37), int64(13)
memory usage: 12.0 MB
```

In [61]:
```
fintech_app_dataset.describe()
```

Out[61]:

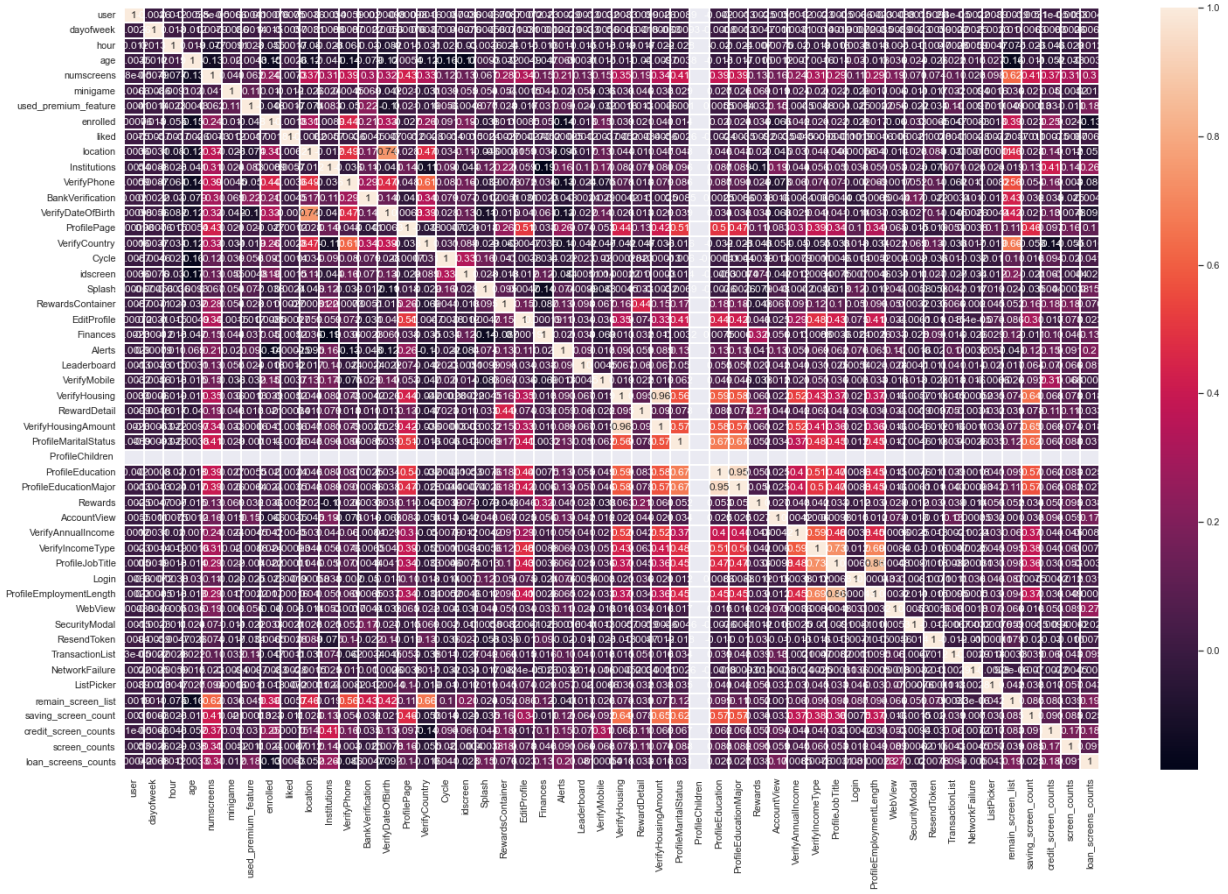
	user	dayofweek	hour	age	numscreens	minigame	used_
count	50000.000000	50000.000000	50000.000000	50000.00000	50000.000000	50000.000000	
mean	186889.729900	3.029860	12.557220	31.72436	21.095900	0.107820	
std	107768.520361	2.031997	7.438072	10.80331	15.728812	0.310156	
min	13.000000	0.000000	0.000000	16.00000	1.000000	0.000000	
25%	93526.750000	1.000000	5.000000	24.00000	10.000000	0.000000	
50%	187193.500000	3.000000	14.000000	29.00000	18.000000	0.000000	
75%	279984.250000	5.000000	19.000000	37.00000	28.000000	0.000000	
max	373662.000000	6.000000	23.000000	101.00000	325.000000	1.000000	

8 rows × 50 columns

Heatmap with Correlation of Fintect_APP_Dataset

In [62]:
```
plt.figure(figsize=(25,16))

sns.heatmap(fintech_app_dataset.corr(), annot=True, linewidths=2)
```

Out[62]: <AxesSubplot:>



In [63]:
```
fintech_app_dataset.columns
```

Out[63]:
```
Index(['user', 'dayofweek', 'hour', 'age', 'numscreens', 'minigame',
       'used_premium_feature', 'enrolled', 'liked', 'location', 'Institutions',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth', 'ProfilePage',
       'VerifyCountry', 'Cycle', 'idscreen', 'Splash', 'RewardsContainer',
       'EditProfile', 'Finances', 'Alerts', 'Leaderboard', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'ProfileEducationMajor', 'Rewards', 'AccountView', 'VerifyAnnualIncome',
       'VerifyIncomeType', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'ResendToken',
       'TransactionList', 'NetworkFailure', 'ListPicker', 'remain_screen_list',
       'saving_screen_count', 'credit_screen_counts', 'screen_counts',
       'loan_screens_counts'],
      dtype='object')
```

In [64]:
```
fintech_app_dataset['ProfileChildren '].unique()
```

Out[64]:
```
array([0])
```

In [65]:
```
corr_matrix = fintech_app_dataset.corr()
corr_matrix['ProfileChildren ']
```

Out[65]:
```
user                        NaN
dayofweek                   NaN
hour                        NaN
age                         NaN
numscreens                  NaN
minigame                    NaN
used_premium_feature        NaN
enrolled                    NaN
liked                       NaN
location                    NaN
Institutions                NaN
VerifyPhone                 NaN
BankVerification            NaN
VerifyDateOfBirth           NaN
ProfilePage                 NaN
VerifyCountry               NaN
Cycle                       NaN
idscreen                    NaN
Splash                      NaN
RewardsContainer            NaN
EditProfile                 NaN
Finances                    NaN
Alerts                      NaN
Leaderboard                 NaN
VerifyMobile                NaN
VerifyHousing               NaN
RewardDetail                NaN
VerifyHousingAmount         NaN
ProfileMaritalStatus        NaN
ProfileChildren             NaN
ProfileEducation            NaN
ProfileEducationMajor       NaN
Rewards                     NaN
AccountView                 NaN
VerifyAnnualIncome          NaN
VerifyIncomeType            NaN
ProfileJobTitle             NaN
Login                       NaN
ProfileEmploymentLength     NaN
WebView                     NaN
SecurityModal               NaN
ResendToken                 NaN
TransactionList             NaN
NetworkFailure              NaN
ListPicker                  NaN
remain_screen_list          NaN
saving_screen_count         NaN
credit_screen_counts        NaN
screen_counts               NaN
loan_screens_counts         NaN
Name: ProfileChildren , dtype: float64
```

In [66]:
```
fintech_app_dataset['ProfileChildren ']
```

Out[66]:
```
0        0
1        0
2        0
3        0
4        0
        ..
49995    0
49996    0
49997    0
49998    0
49999    0
Name: ProfileChildren , Length: 50000, dtype: int32
```

Data Preprocessing

Split Dataset Into Train and Test

In [67]:
```
clean_fintech_app_dataset = fintech_app_dataset
target = fintech_app_dataset['enrolled']
fintech_app_dataset.drop(columns='enrolled', inplace=True)
```

In [68]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train,Y_test = train_test_split(fintech_app_dataset, target, t
```

In [69]:
```
print('Shape of X_train', X_train.shape)
print('Shape of X_test ', X_test.shape)
print('Shape of Y_train', Y_train.shape)
print('Shape of Y_test', Y_test.shape)
```

```
Shape of X_train (40000, 49)
Shape of X_test  (10000, 49)
Shape of Y_train (40000,)
Shape of Y_test (10000,)
```

Take User ID in Another Variable

In [70]:
```
train_user_id = X_train['user']
X_train.drop(columns='user', inplace=True)
test_user_id = X_test['user']
X_test.drop(columns='user', inplace=True)
```

In [71]:
```
print('Shape of X_train', X_train.shape)
print('Shape of X_test ', X_test.shape)
print('Shape of Train_User_Id', train_user_id.shape)
print('Shape of Test_User_ID', test_user_id.shape)
```

```
Shape of X_train (40000, 48)
Shape of X_test  (10000, 48)
Shape of Train_User_Id (40000,)
Shape of Test_User_ID (10000,)
```

Feature Scaling

In [72]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_sc = sc.fit_transform(X_train)
X_test_sc = sc.transform(X_test)
```

Model Building

In [73]:
```
# Import Required Packages For Buiding Model Effectively and Easily

from sklearn.metrics import confusion_matrix, classification_report, accuracy_sc
```

1- Decision Tree

In [74]:
```
# Without Standered Scaling Dataset

from sklearn.tree import DecisionTreeClassifier
decison_tree_model = DecisionTreeClassifier(criterion='entropy', random_state=0)
decison_tree_model.fit(X_train, Y_train)
y_pred_decision_tree = decison_tree_model.predict(X_test)

accuracy_score(Y_test, y_pred_decision_tree)
```

Out[74]: 0.6936

In [75]:
```
# With Standered Scaling Dataset

from sklearn.tree import DecisionTreeClassifier
decison_tree_model_with_standered_scaling = DecisionTreeClassifier(criterion='ent
decison_tree_model_with_standered_scaling.fit(X_train_sc,  Y_train)
y_pred_decision_tree_with_scaling = decison_tree_model_with_standered_scaling.pre

accuracy_score(Y_test,  y_pred_decision_tree_with_scaling)
```

Out[75]: 0.6932

2-KNN (K-Nearest-Neighbors)

In [76]:
```
# Without Standered Scaling Dataset

from sklearn.neighbors import  KNeighborsClassifier
k_neighbores_classifier_model = KNeighborsClassifier(n_neighbors=5, metric='minko
k_neighbores_classifier_model.fit(X_train,Y_train)
y_pred_k_neighbors_classifier =k_neighbores_classifier_model.predict(X_test)

accuracy_score(Y_test, y_pred_k_neighbors_classifier)
```

Out[76]: 0.6978

In [77]:
```
# With Standered Scaling Dataset
from sklearn.neighbors import KNeighborsClassifier
k_neighbores_classifier_model_with_scaling = KNeighborsClassifier(n_neighbors=5,
k_neighbores_classifier_model_with_scaling.fit(X_train_sc, Y_train)
y_pred_k_neighbores_classifiers_with_scaling = k_neighbores_classifier_model_with

accuracy_score(Y_test, y_pred_k_neighbores_classifiers_with_scaling)
```

Out[77]: 0.7314

3-Naive Bayes

In [78]:
```
# Without Standered Scaling Dataset
from sklearn.naive_bayes import GaussianNB
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, Y_train)
y_pred_naive_bayes_model = naive_bayes_model.predict(X_test)

accuracy_score(Y_test, y_pred_naive_bayes_model)
```

Out[78]: 0.7114

In [79]:
```
# With Standered Scaling Dataset
from sklearn.naive_bayes import GaussianNB
naive_bayes_model_with_scaling = GaussianNB()
naive_bayes_model_with_scaling.fit(X_train_sc, Y_train)
y_pred_naive_bayes_model_with_scaling = naive_bayes_model_with_scaling.predict(X_

accuracy_score(Y_test, y_pred_naive_bayes_model_with_scaling)
```

Out[79]: 0.7114

4- Random Forest

In [80]:
```
# Without Standered Scaling Dataset
from sklearn.ensemble import RandomForestClassifier
random_forest_classifer_model = RandomForestClassifier(n_estimators=10, criterion
random_forest_classifer_model.fit(X_train,Y_train)
y_pred_random_forest_classifier_model = random_forest_classifer_model.predict(X_t

accuracy_score(Y_test, y_pred_random_forest_classifier_model)
```

Out[80]: 0.7621

In [81]:
```
# With Standered Scaling Dataset
from sklearn.ensemble import RandomForestClassifier
random_forest_classifer_model_with_scaling =  RandomForestClassifier(n_estimators
random_forest_classifer_model_with_scaling.fit(X_train_sc, Y_train)
y_pred_random_forest_classifier_model_with_scaling = random_forest_classifer_mode

accuracy_score(Y_test, y_pred_random_forest_classifier_model_with_scaling)
```

Out[81]: 0.7616

5-Logistic Regression

In [82]:
```
# Without Standered Scaling Dataset
from sklearn.linear_model import LogisticRegression
logistic_regression_model = LogisticRegression(random_state=0, penalty = 'l2')
logistic_regression_model.fit(X_train, Y_train)
y_pred_logistic_regression_model = logistic_regression_model.predict(X_test)

accuracy_score(Y_test, y_pred_logistic_regression_model)
```

```
D:\Annaconda\lib\site-packages\sklearn\linear_model\_logistic.py:444: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[82]: 0.7686

In [83]:
```
# With Standered Scaling Dataset
from sklearn.linear_model import LogisticRegression
logistic_regression_model_with_scaling = LogisticRegression(random_state=0, penal
logistic_regression_model_with_scaling.fit(X_train_sc,Y_train)
y_pred_logistic_regression_model_with_scaling = logistic_regression_model_with_sc

accuracy_score(Y_test, y_pred_logistic_regression_model_with_scaling)
```

Out[83]: 0.768

6-Support Vector Machine

In [84]:
```
# Without Standered Scaling Dataset

from sklearn.svm import SVC
support_vector_machine_model = SVC()
support_vector_machine_model.fit(X_train, Y_train)
y_pred_support_vector_machine_model = support_vector_machine_model.predict(X_test

accuracy_score(Y_test, y_pred_support_vector_machine_model)
```

Out[84]: 0.7609

In [85]:
```
# With Standered Scaling Dataset

from sklearn.svm import SVC
support_vector_machine_model_with_scaling = SVC()
support_vector_machine_model_with_scaling.fit(X_train_sc,Y_train)
y_pred_support_vector_machine_model_with_scaling = support_vector_machine_model_v

accuracy_score(Y_test, y_pred_support_vector_machine_model_with_scaling)
```

Out[85]: 0.7789

7- XGBoost (Extreme Grading Boosting)

In [86]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in d:\annaconda\lib\site-packages (1.6.
2)
Requirement already satisfied: numpy in d:\annaconda\lib\site-packages (from xg
boost) (1.21.5)
Requirement already satisfied: scipy in d:\annaconda\lib\site-packages (from xg
boost) (1.7.3)
```

In [87]:
```
# Without Standered Scaling Dataset

from xgboost import XGBClassifier
xg_boost_model= XGBClassifier()
xg_boost_model.fit(X_train, Y_train)
y_pred_xg_boost_model = xg_boost_model.predict(X_test)

accuracy_score(Y_test, y_pred_xg_boost_model)
```

Out[87]: 0.781

In [88]:
```
# With Standered Scaling Dataset

from xgboost import XGBClassifier
xg_boost_model_with_scaling = XGBClassifier()
xg_boost_model_with_scaling.fit(X_train_sc, Y_train)
y_pred_xg_boost_model_with_scaling = xg_boost_model_with_scaling.predict(X_test_s

accuracy_score(Y_test, y_pred_xg_boost_model_with_scaling)
```

Out[88]: 0.781

In [89]:
```
# XGB Classifier With Parameter Tuning and Without Standered Scaling Dataset

xgb_model = XGBClassifier(
    learning_rate=0.01,
    n_estimators=5000,
    max_depth=4,
    min_child_weight=6,
    gamma=0,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0.005,
    objective = 'binary:logistic',
    nthread=4,
    scale_pos_weight=1,
    seed=27
)

xgb_model.fit(X_train,Y_train)
y_pred_xgb_model = xg_boost_model.predict(X_test)

accuracy_score(Y_test, y_pred_xgb_model)
```

Out[89]: 0.781

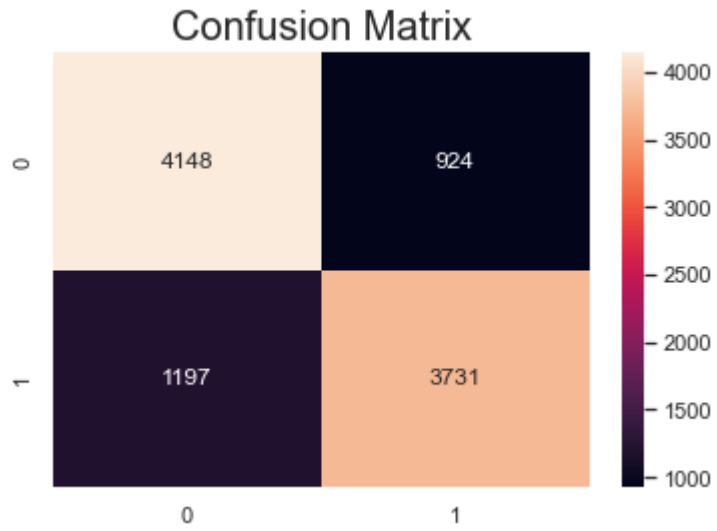
In [90]:
```
#XGB Classifer With parameter Tuning and With Standered Scaling Dataset

xgb_model_with_scaling = XGBClassifier(
    learning_rate=0.01,
    n_estimators=5000,
    max_depth=4,
    min_child_weight=6,
    gamma=0,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0.005,
    objective='binary:logistic',
    nthread=4,
    scale_pos_weight=1,
    seed=27
)

xgb_model_with_scaling.fit(X_train_sc,Y_train)
y_pred_xgb_model_with_scaling = xgb_model_with_scaling.predict(X_test_sc)

accuracy_score(Y_test,y_pred_xgb_model_with_scaling)
```

Out[90]:
```
0.7879
```

Confusion Matrix

In [91]:
```
cm_xgb_pt_2 = confusion_matrix(Y_test,y_pred_xgb_model_with_scaling)
sns.heatmap(cm_xgb_pt_2,annot=True, fmt='g')
plt.title('Confusion Matrix', fontsize=20)
```

Out[91]:
```
Text(0.5, 1.0, 'Confusion Matrix')
```



Classification Report

In [92]:
```
classification_report_xgb_point_2 = classification_report(Y_test, y_pred_xgb_mode
print('Classification Report', classification_report_xgb_point_2)
```

```
Classification Report              precision    recall  f1-score   support

           0       0.78      0.82      0.80      5072
           1       0.80      0.76      0.78      4928

    accuracy                           0.79     10000
   macro avg       0.79      0.79      0.79     10000
weighted avg       0.79      0.79      0.79     10000
```

Cross Validation

In [93]:
```
from sklearn.model_selection import cross_val_score
cross_validation = cross_val_score(estimator=xg_boost_model_with_scaling,X=X_trai
print('Cross Validation in XGBoost Model', cross_validation)
print('Cross Validation in XGBoost Model in Mean', cross_validation.mean())
```

```
Cross Validation in XGBoost Model [0.78725 0.773   0.785   0.78125 0.7785  0.78
375 0.788   0.7805  0.792
 0.7755 ]
Cross Validation in XGBoost Model in Mean 0.782475
```

Mapping Predicted Output To Target

In [94]:
```
final_result = pd.concat([test_user_id,Y_test], axis=1)
final_result['predicted result'] = y_pred_xgb_model_with_scaling

final_result
```

Out[94]:

	user	enrolled	predicted result
11841	239786	1	1
19602	279644	1	1
45519	98290	0	0
25747	170150	1	1
42642	237568	1	0
...	...	...	...
25091	143036	1	1
27853	91158	1	1
47278	248318	0	0
37020	142418	1	1
2217	279355	1	0

10000 rows × 3 columns

Save The Model

1- Using Pickle

In [95]:
```
import pickle

#Save Model
pickle.dump(xgb_model_with_scaling, open('Fintech_APP_ML_Model.pickle', 'wb'))

#Load Model
ml_model_pickle = pickle.load(open('Fintech_APP_ML_Model.pickle', 'rb'))

#Predict The Output

y_pred_ml = ml_model_pickle.predict(X_test_sc)

#Confusion Matrix

confusion_matrix_pickle = confusion_matrix(Y_test, y_pred_ml)
print('Confusion matrix  \n = ', confusion_matrix_pickle)

# Show Accuracy


print('Accuracy of The Model is \n =', accuracy_score(Y_test, y_pred_ml))
```

```
Confusion matrix
 =  [[4148  924]
 [1197 3731]]
Accuracy of The Model is
 = 0.7879
```

2-Using Joblib

In [96]:
```
pip install joblib
```

```
Requirement already satisfied: joblib in d:\annaconda\lib\site-packages (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [97]:
```
import joblib

#Save The Model

joblib.dump(xgb_model_with_scaling,'Fintech_App_ML_Model.joblib')

#Load The Model

load_model_joblib = joblib.load('Fintech_App_ML_Model.joblib')

# Predict The Model

y_pred_ml_joblib = load_model_joblib.predict(X_test_sc)

#Confusion Matrix

confusion_matrix_joblib = confusion_matrix(Y_test, y_pred_ml_joblib)
print('Confusion Matrix \n =', confusion_matrix_joblib)

# Show Accuracy

print('Accuracy Score Of The Model is \n =', accuracy_score(Y_test, y_pred_ml_jo
```

```
Confusion Matrix
 = [[4148  924]
 [1197 3731]]
Accuracy Score Of The Model is
 = 0.7879
```

In [ ]: