ExtraaLearn Project

Context

The EdTech industry has been surging in the past decade immensely, and according to a forecast, the Online Education market would be worth $286.62bn by 2023 with a compound annual growth rate (CAGR) of 10.26% from 2018 to 2023. The modern era of online education has enforced a lot in its growth and expansion beyond any limit. Due to having many dominant features like ease of information sharing, personalized learning experience, transparency of assessment, etc, it is now preferable to traditional education.

In the present scenario due to the Covid-19, the online education sector has witnessed rapid growth and is attracting a lot of new customers. Due to this rapid growth, many new companies have emerged in this industry. With the availability and ease of use of digital marketing resources, companies can reach out to a wider audience with their offerings. The customers who show interest in these offerings are termed as leads. There are various sources of obtaining leads for Edtech companies, like

- The customer interacts with the marketing front on social media or other online platforms.
- The customer browses the website/app and downloads the brochure
- The customer connects through emails for more information.

The company then nurtures these leads and tries to convert them to paid customers. For this, the representative from the organization connects with the lead on call or through email to share further details.

Objective

ExtraaLearn is an initial stage startup that offers programs on cutting-edge technologies to students and professionals to help them upskill/reskill. With a large number of leads being generated on a regular basis, one of the issues faced by ExtraaLearn is to identify which of the leads are more likely to convert so that they can allocate resources accordingly. You, as a data scientist at ExtraaLearn, have been provided the leads data to:

- Analyze and build an ML model to help identify which leads are more likely to convert to paid customers,
- Find the factors driving the lead conversion process
- Create a profile of the leads which are likely to convert

Data Description

The data contains the different attributes of leads and their interaction details with ExtraaLearn. The detailed data dictionary is given below.

Data Dictionary

- ID: ID of the lead
- age: Age of the lead
- current_occupation: Current occupation of the lead. Values include 'Professional','Unemployed',and 'Student'
- first_interaction: How did the lead first interacted with ExtraaLearn. Values include 'Website', 'Mobile App'
- profile_completed: What percentage of profile has been filled by the lead on the website/mobile app. Values include Low - (0-50%), Medium - (50-75%), High (75-100%)
- website_visits: How many times has a lead visited the website
- time_spent_on_website: Total time spent on the website
- page_views_per_visit: Average number of pages on the website viewed during the visits.
- last_activity: Last interaction between the lead and ExtraaLearn.
    - Email Activity: Seeking for details about program through email, Representative shared information with lead like brochure of program , etc
    - Phone Activity: Had a Phone Conversation with representative, Had conversation over SMS with representative, etc
    - Website Activity: Interacted on live chat with representative, Updated profile on website, etc
- print_media_type1: Flag indicating whether the lead had seen the ad of ExtraaLearn in the Newspaper.
- print_media_type2: Flag indicating whether the lead had seen the ad of ExtraaLearn in the Magazine.
- digital_media: Flag indicating whether the lead had seen the ad of ExtraaLearn on the digital platforms.
- educational_channels: Flag indicating whether the lead had heard about ExtraaLearn in the education channels like online forums, discussion threads, educational websites, etc.
- referral: Flag indicating whether the lead had heard about ExtraaLearn through reference.
- status: Flag indicating whether the lead was converted to a paid customer or not.

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing necessary libraries and data

In [2]:
```
# Importing the basic libraries we will require for the project

# Data Readers
import pandas as pd
```

```
import numpy as np

# Data visualization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import plotly.express as px

# Machine Learning models from Scikit-Learn
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier

# Other functions from Scikit-Learn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# Get diferent metric scores
from sklearn.metrics import confusion_matrix,classification_report,roc_auc_score,preci
from sklearn.metrics import accuracy_score
# Ignore warnings from function usage
import warnings;
import numpy as np
warnings.filterwarnings('ignore')
```

Data Overview

- Observations
- Sanity checks

In [3]:
```
#Load Data Set
df=pd.read_csv('/content/drive/MyDrive/Classification and Hypothesis Testing Project/E

#Start review of Data
#Show first 5 rows of data
df.head()
```

Out[3]:

	ID	age	current_occupation	first_interaction	profile_completed	website_visits	time_spent_on_
0	EXT001	57	Unemployed	Website	High	7	
1	EXT002	56	Professional	Mobile App	Medium	2	
2	EXT003	52	Professional	Website	Medium	3	
3	EXT004	53	Unemployed	Website	High	4	
4	EXT005	23	Student	Website	High	4	

In [4]:

```
#Show last 5 rows of data
df.tail()
```

Out[4]:

	ID	age	current_occupation	first_interaction	profile_completed	website_visits	time_spent
4607	EXT4608	35	Unemployed	Mobile App	Medium	15	
4608	EXT4609	55	Professional	Mobile App	Medium	8	
4609	EXT4610	58	Professional	Website	High	2	
4610	EXT4611	57	Professional	Mobile App	Medium	1	
4611	EXT4612	55	Professional	Website	Medium	4	

In [5]:

```
# Compare data shape to what I see.  I should have 4612 rows and 15 columns
df.shape
```

Out[5]:

```
(4612, 15)
```

In [6]:

```
#Check Data Types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4612 entries, 0 to 4611
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   ID                     4612 non-null   object 
 1   age                    4612 non-null   int64  
 2   current_occupation     4612 non-null   object 
 3   first_interaction      4612 non-null   object 
 4   profile_completed      4612 non-null   object 
 5   website_visits         4612 non-null   int64  
 6   time_spent_on_website  4612 non-null   int64  
 7   page_views_per_visit   4612 non-null   float64
 8   last_activity          4612 non-null   object 
 9   print_media_type1      4612 non-null   object 
 10  print_media_type2      4612 non-null   object 
 11  digital_media          4612 non-null   object 
 12  educational_channels   4612 non-null   object 
 13  referral               4612 non-null   object 
 14  status                 4612 non-null   int64  
dtypes: float64(1), int64(4), object(10)
memory usage: 540.6+ KB
```

In [7]:
```
#data looks good from standpoint of no missing values and data types are assigned prop
#check to make sure that all customer ID's are unique and that binary objects only hav
df.nunique()
```

Out[7]:
```
ID                       4612
age                        46
current_occupation          3
first_interaction           2
profile_completed           3
website_visits             27
time_spent_on_website    1623
page_views_per_visit     2414
last_activity               3
print_media_type1           2
print_media_type2           2
digital_media               2
educational_channels        2
referral                    2
status                      2
dtype: int64
```

In [9]:
```
#Drop ID Column
df=df.drop(['ID'], axis=1)
```

In [10]:
```
#Check the count of each unique category in each of the categorical variable
#Create list of numerical features
num_col=['age', 'website_visits', 'time_spent_on_website', 'page_views_per_visit']

#Create list of categorical features
category_col=['current_occupation', 'first_interaction', 'profile_completed', 'last_ac
```

In [11]:
```
#Convert data type for each categorical variable to 'category'
for column in category_col:
  df[column]=df[column].astype('category')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4612 entries, 0 to 4611
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype   
---  ------                 --------------  -----   
 0   age                    4612 non-null   int64   
 1   current_occupation     4612 non-null   category
 2   first_interaction      4612 non-null   category
 3   profile_completed      4612 non-null   category
 4   website_visits         4612 non-null   int64   
 5   time_spent_on_website  4612 non-null   int64   
 6   page_views_per_visit   4612 non-null   float64 
 7   last_activity          4612 non-null   category
 8   print_media_type1      4612 non-null   category
 9   print_media_type2      4612 non-null   category
 10  digital_media          4612 non-null   category
 11  educational_channels   4612 non-null   category
 12  referral               4612 non-null   category
 13  status                 4612 non-null   category
dtypes: category(10), float64(1), int64(3)
memory usage: 190.5 KB
```

In [12]:

```
#provide information regarding

df[num_col].describe()
```

Out[12]:

	age	website_visits	time_spent_on_website	page_views_per_visit
count	4612.000000	4612.000000	4612.000000	4612.000000
mean	46.201214	3.566782	724.011275	3.026126
std	13.161454	2.829134	743.828683	1.968125
min	18.000000	0.000000	0.000000	0.000000
25%	36.000000	2.000000	148.750000	2.077750
50%	51.000000	3.000000	376.000000	2.792000
75%	57.000000	5.000000	1336.750000	3.756250
max	63.000000	30.000000	2537.000000	18.434000

In [13]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4612 entries, 0 to 4611
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype   
---  ------                 --------------  -----   
 0   age                    4612 non-null   int64   
 1   current_occupation     4612 non-null   category
 2   first_interaction      4612 non-null   category
 3   profile_completed      4612 non-null   category
 4   website_visits         4612 non-null   int64   
 5   time_spent_on_website  4612 non-null   int64   
 6   page_views_per_visit   4612 non-null   float64 
 7   last_activity          4612 non-null   category
 8   print_media_type1      4612 non-null   category
 9   print_media_type2      4612 non-null   category
 10  digital_media          4612 non-null   category
 11  educational_channels   4612 non-null   category
 12  referral               4612 non-null   category
 13  status                 4612 non-null   category
dtypes: category(10), float64(1), int64(3)
memory usage: 190.5 KB
```

Exploratory Data Analysis (EDA)

- EDA is an important part of any project involving data.
- It is important to investigate and understand the data better before building a model with it.
- A few questions have been mentioned below which will help you approach the analysis in the right manner and generate insights from the data.
- A thorough analysis of the data, in addition to the questions mentioned below, should be done.

Questions

1. Leads will have different expectations from the outcome of the course and the current occupation may play a key role in getting them to participate in the program. Find out how current occupation affects lead status.
2. The company's first impression on the customer must have an impact. Do the first channels of interaction have an impact on the lead status?
3. The company uses multiple modes to interact with prospects. Which way of interaction works best?
4. The company gets leads from various channels such as print media, digital media, referrals, etc. Which of these channels have the highest lead conversion rate?
5. People browsing the website or mobile application are generally required to create a profile by sharing their personal data before they can access additional information.Does having more details about a prospect increase the chances of conversion?

Data Preprocessing

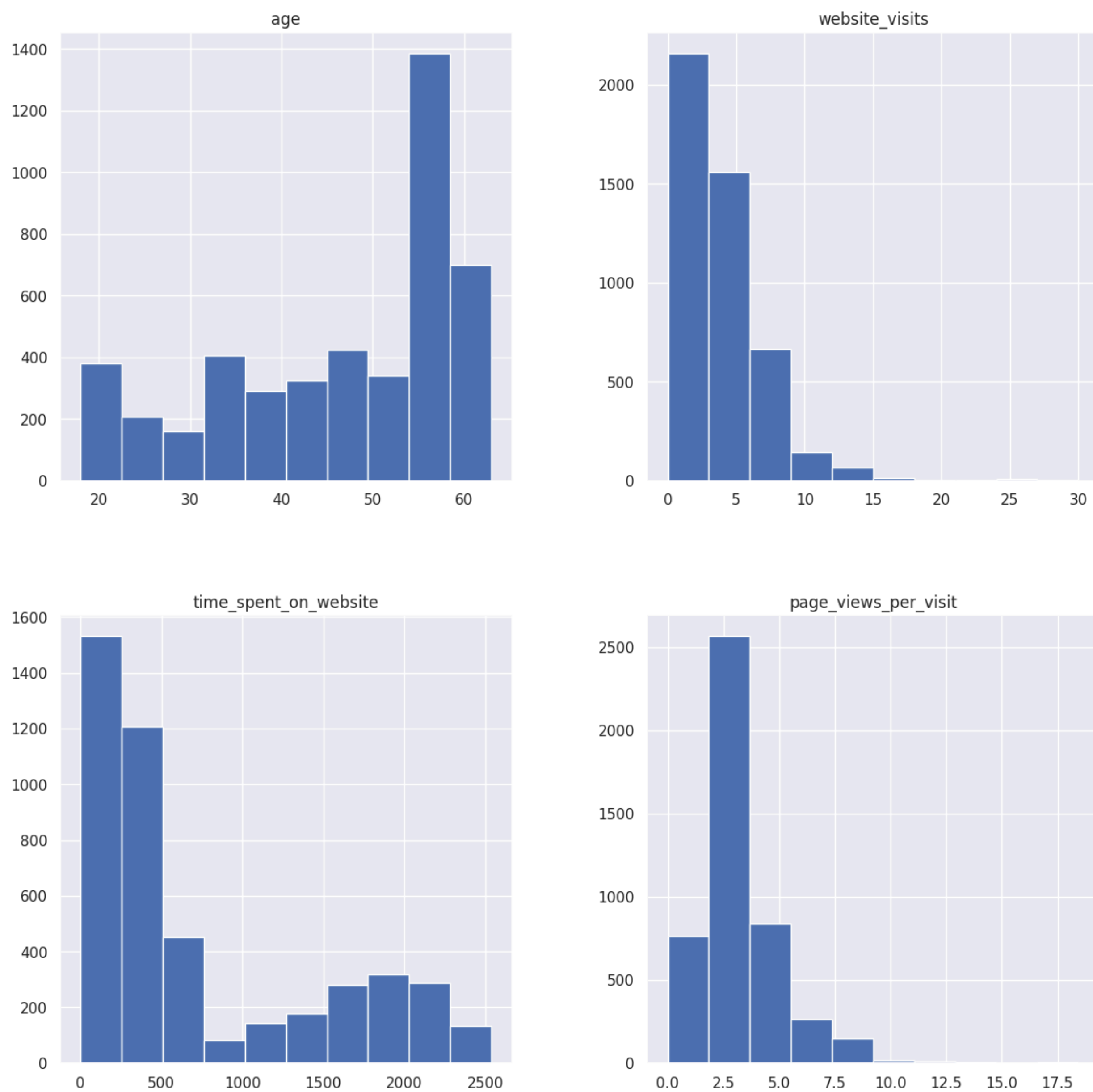
- Missing value treatment (if needed)
- Feature engineering (if needed)
- Outlier detection and treatment (if needed)
- Preparing data for modeling
- Any other preprocessing steps (if needed)

In [ ]:
```
# Completed above
```

EDA

- It is a good idea to explore the data once again after manipulating it.

In [ ]:
```
#Creating histograms
df[num_col].hist(figsize=(14,14))
plt.show()
```



We see a spike in the number of people spending 0 time on the phone as we trend below 700 minutes. Above 700 minutes, there appers to be some normalization of the curve. I am assuming that the spike towards 0 is representative of those using other channels such as the phone or email. This also may represent that there is a minimum amount of time required to spend on the website to create a sale at around 1000 minutes.

In [14]:
```
#Find if time spent on website less that 700 has many sales
df.query('time_spent_on_website < 700 and status==1')
```

Out[14]:

	age	current_occupation	first_interaction	profile_completed	website_visits	time_spent_on_websi
3	53	Unemployed	Website	High	4	4
6	56	Professional	Mobile App	Medium	13	6
10	52	Professional	Website	Medium	2	4
11	57	Professional	Website	High	3	6
34	55	Professional	Website	High	2	5
...	...	...	...	...	...	
4555	58	Unemployed	Website	High	4	5
4574	62	Unemployed	Website	High	5	4
4578	58	Professional	Website	High	0	
4604	58	Professional	Website	Medium	2	5
4609	58	Professional	Website	High	2	2

663 rows × 14 columns

In [15]:
```
# find if time spent on website greater than 700 has many sales
df.query('time_spent_on_website > 700 and status==1')
```

Out[15]:

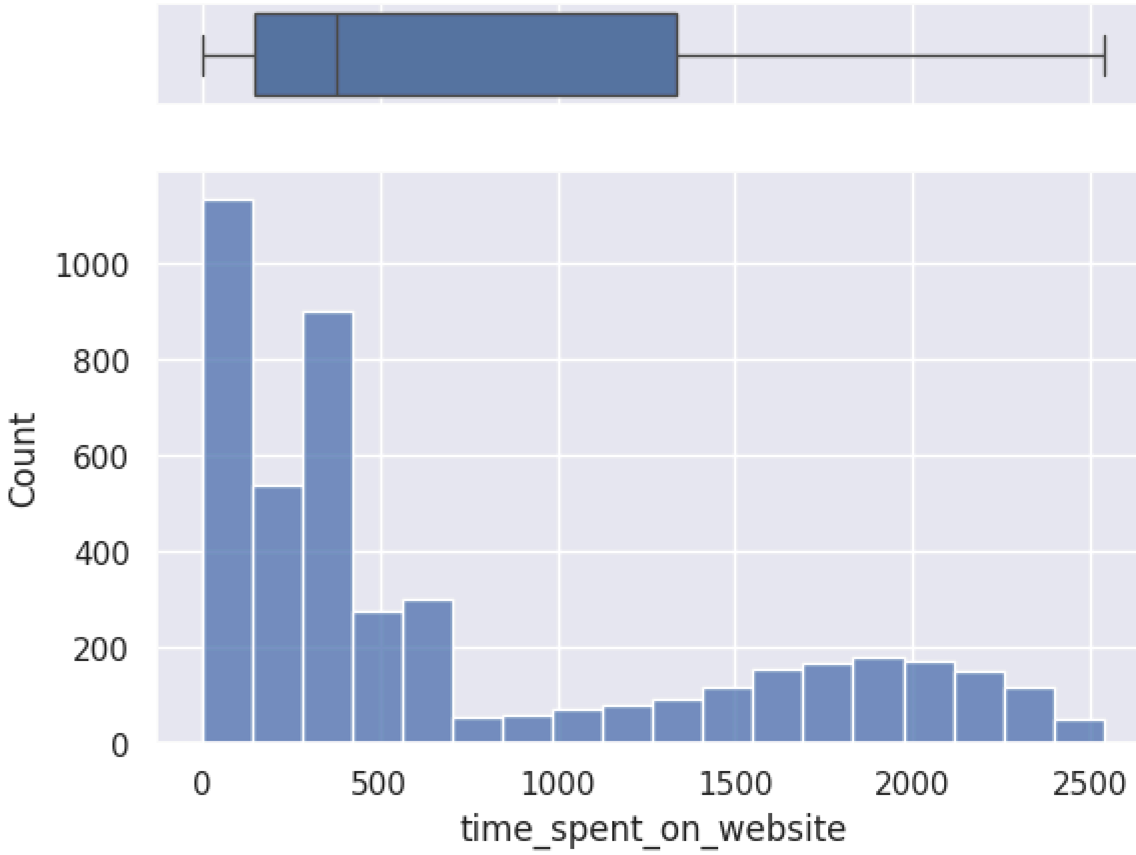
	age	current_occupation	first_interaction	profile_completed	website_visits	time_spent_on_websi
0	57	Unemployed	Website	High	7	16
8	57	Professional	Mobile App	High	2	22
16	47	Professional	Website	High	3	14
24	49	Professional	Mobile App	High	5	22
25	46	Professional	Website	Medium	4	18
...	...	...	...	...	...	
4571	54	Professional	Website	High	12	15
4580	55	Professional	Website	Medium	3	24
4583	49	Professional	Website	Medium	24	10
4587	60	Professional	Website	High	2	15
4588	60	Professional	Mobile App	Medium	1	21

713 rows × 14 columns

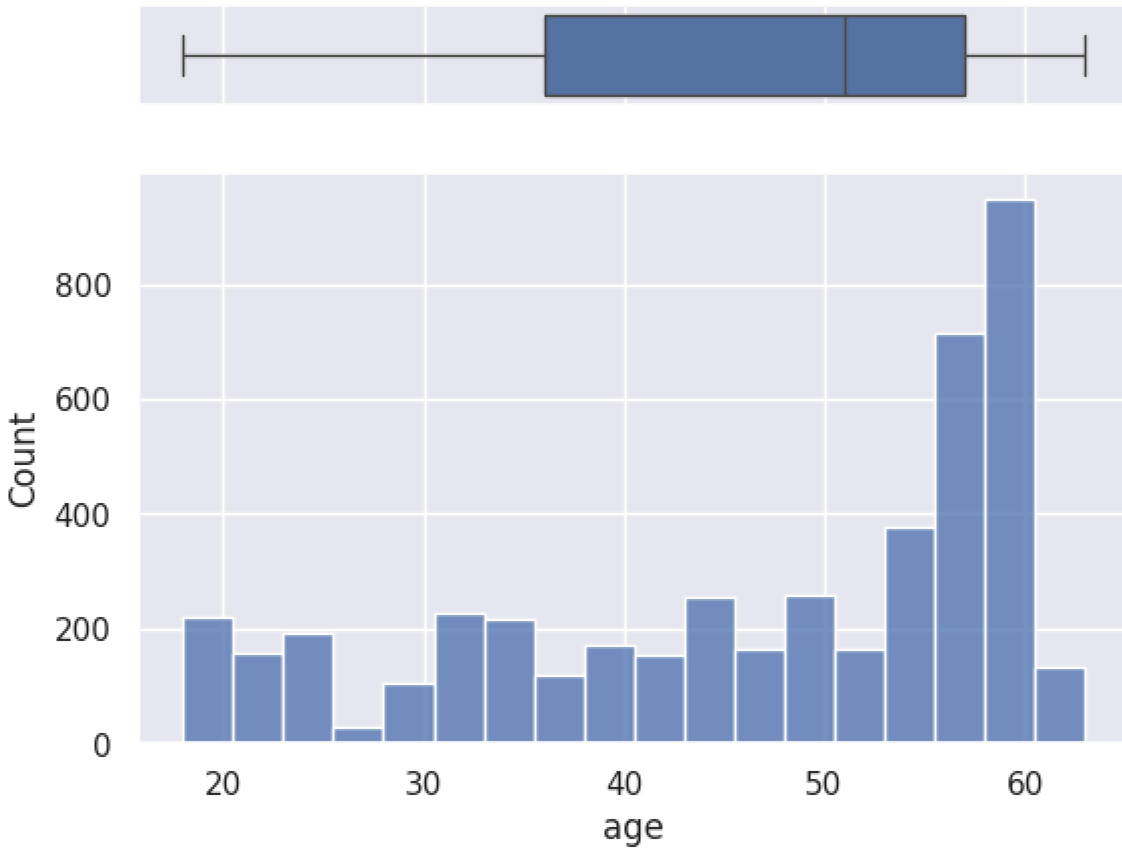
Here, we determined that time spent on the website did not have much effect on sales across the board. 663 sales below 700 minutes and 713 sales above 70 minutes on the site is what was observed. That is not a large enough variance in my opinion to call it relevant. To me this states that our other methods of communication are generating just as many sales as the website, making it unaffecting in the big picture.

In [16]:

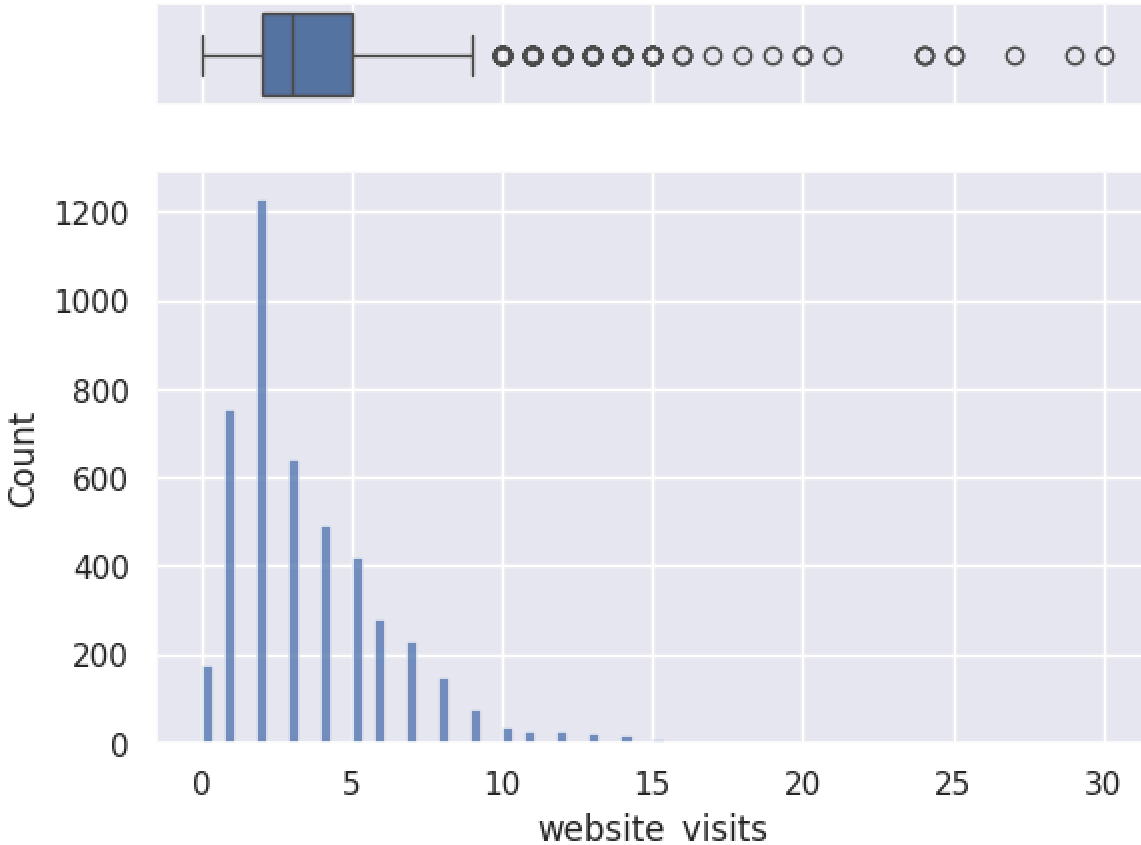
```
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={"height_ratios": (.15
sns.boxplot(df["time_spent_on_website"], orient="h", ax=ax_box)
sns.histplot(data=df, x="time_spent_on_website", ax=ax_hist)
ax_box.set(xlabel='')
plt.show()
```



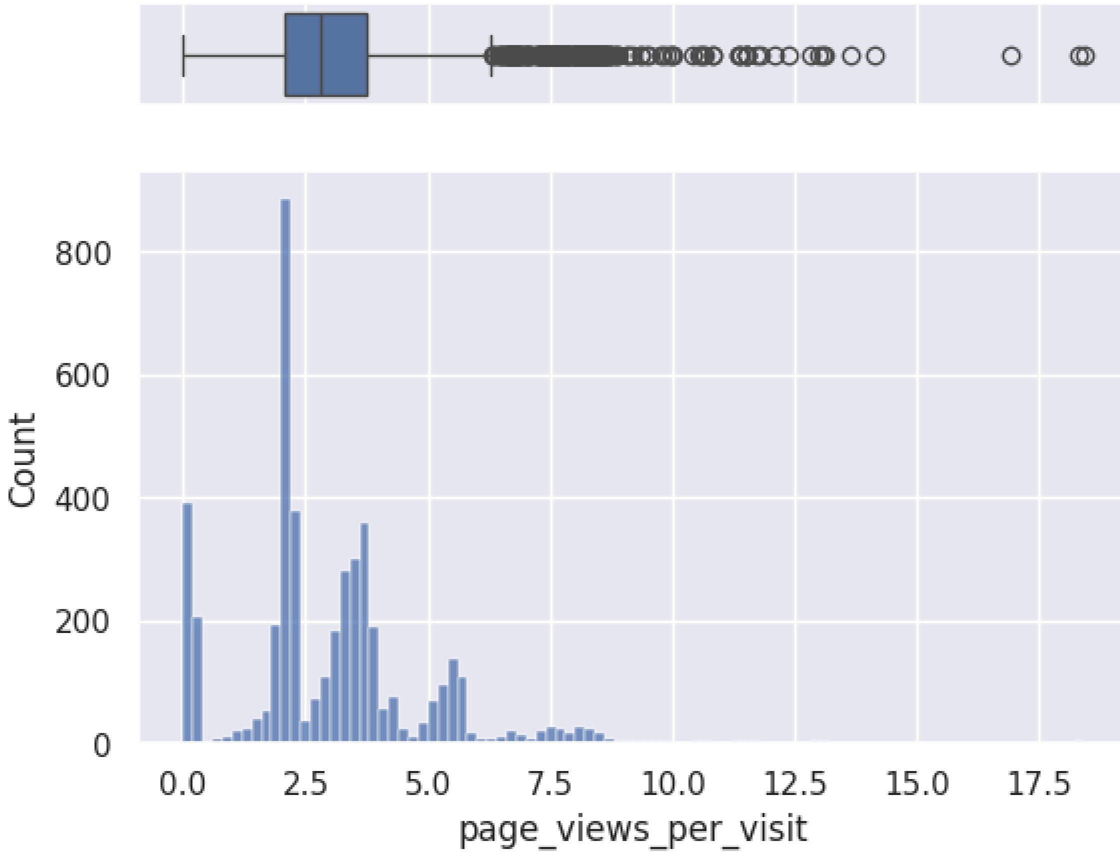
In [17]:
```
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={"height_ratios": (.15
sns.boxplot(df["age"], orient="h", ax=ax_box)
sns.histplot(data=df, x="age", ax=ax_hist)
ax_box.set(xlabel='')
plt.show()
```



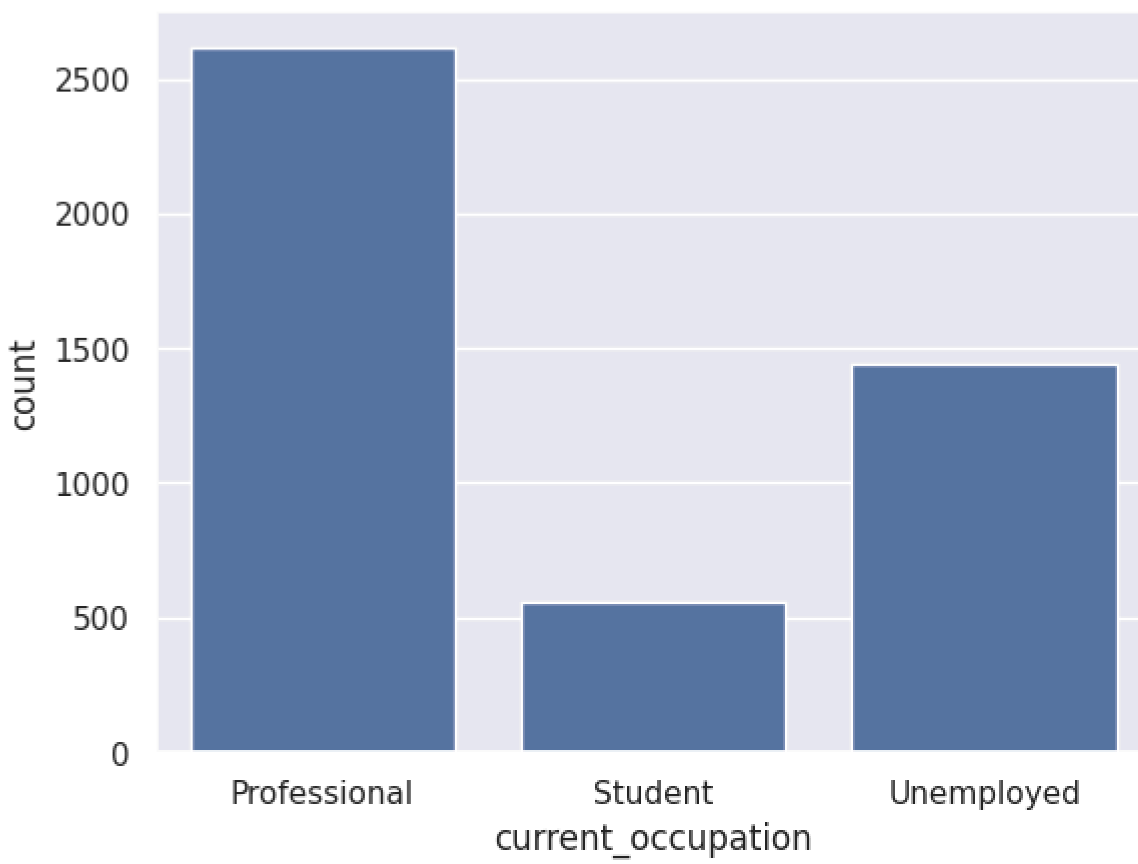
In [18]:
```
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={"height_ratios": (.15
sns.boxplot(df["website_visits"], orient="h", ax=ax_box)
sns.histplot(data=df, x="website_visits", ax=ax_hist)
ax_box.set(xlabel='')
plt.show()
```



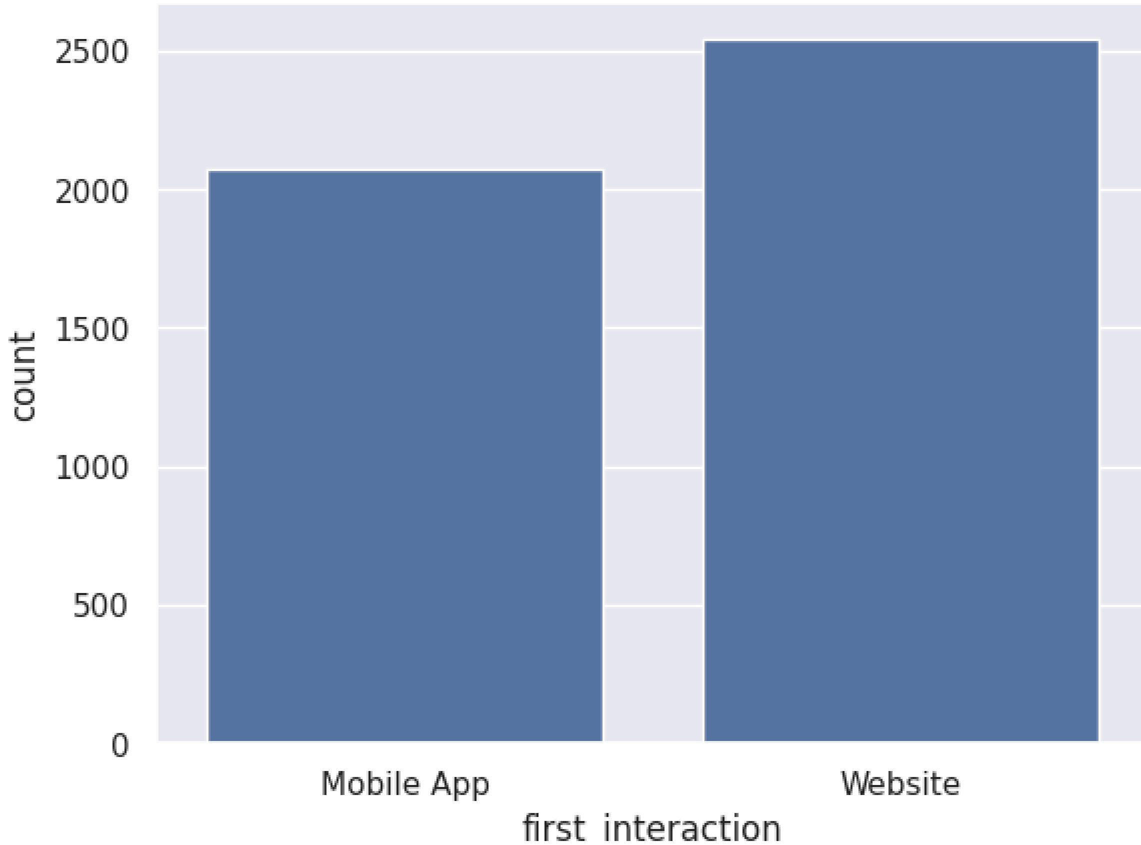
In [19]:
```
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={"height_ratios": (.15
sns.boxplot(df["page_views_per_visit"], orient="h", ax=ax_box)
sns.histplot(data=df, x="page_views_per_visit", ax=ax_hist)
ax_box.set(xlabel='')
plt.show()
```



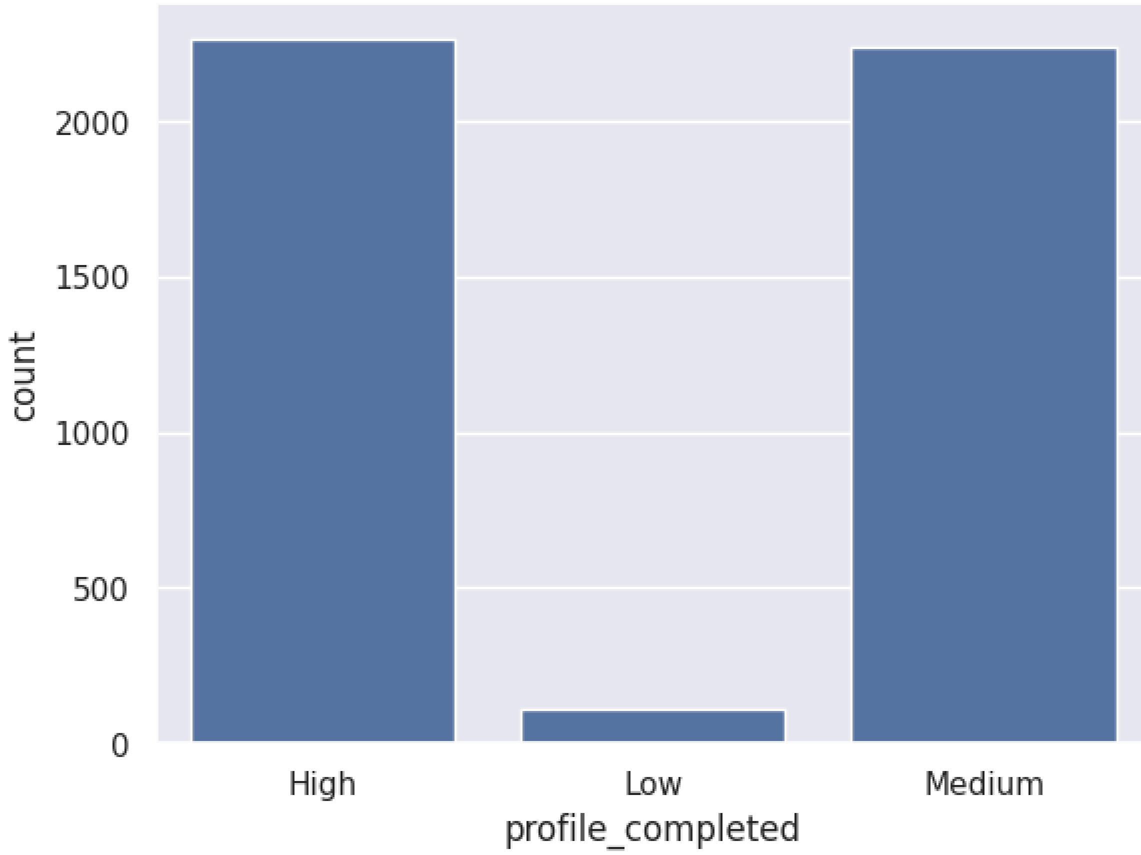
In [21]:
```
sns.countplot(x=df['current_occupation'])
plt.show()
```



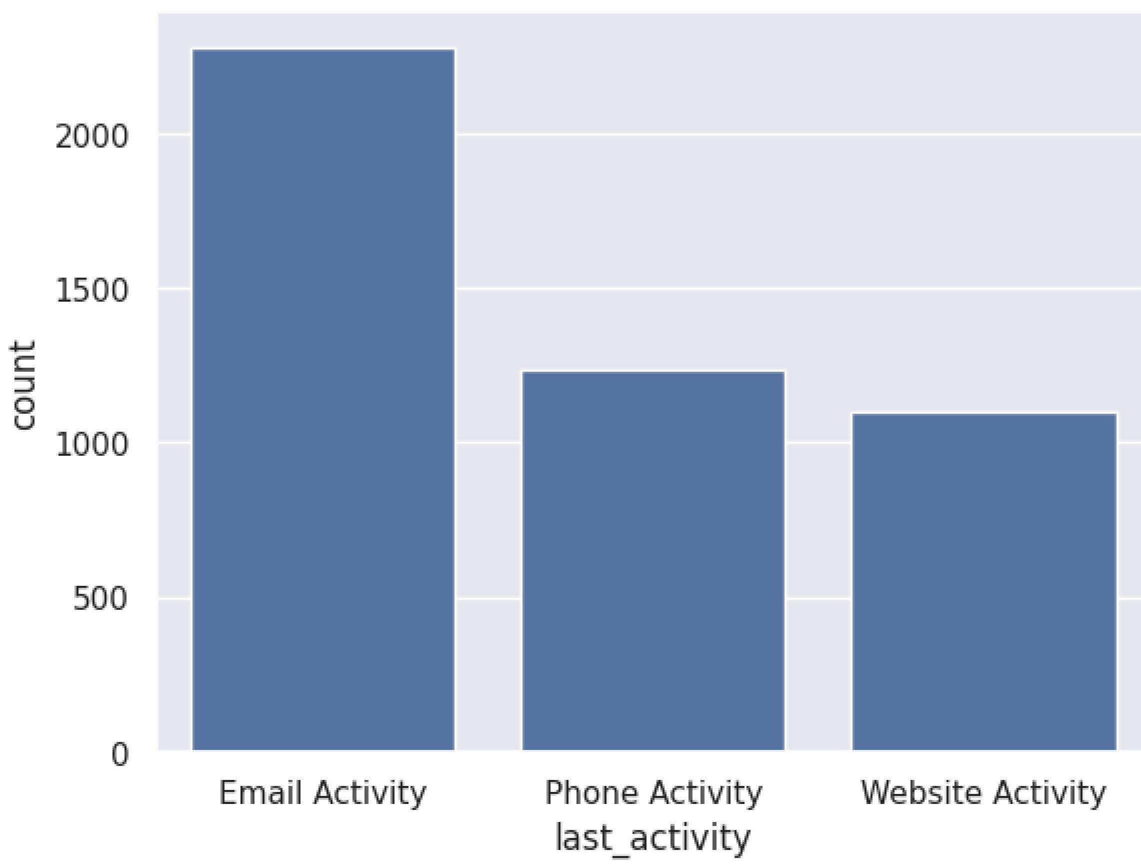
In [20]:
```
sns.countplot(x=df['first_interaction'])
plt.show()
```



In [22]:
```
sns.countplot(x=df['profile_completed'])
plt.show()
```



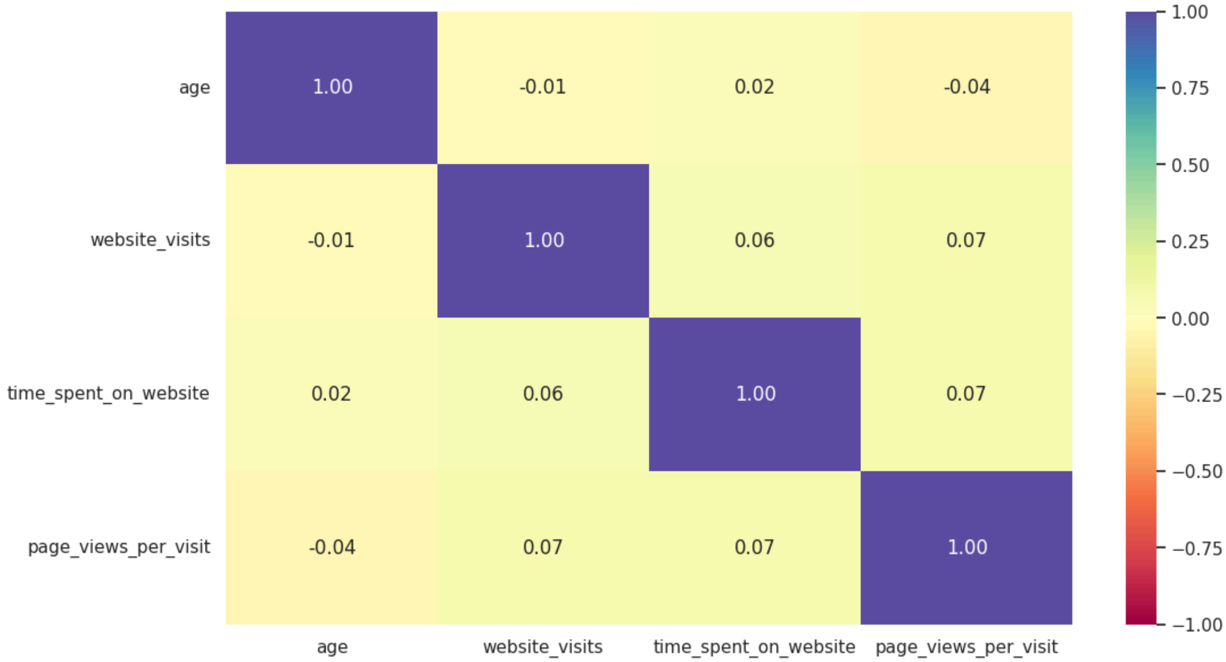
In [23]:
```
sns.countplot(x=df['last_activity'])
plt.show()
```



In [24]:
```
### print the % sub categories of each category
for i in category_col:
  print(df[i].value_counts(normalize=True))
  print('*'*40)
```

```
current_occupation
Professional    0.567216
Unemployed      0.312446
Student         0.120338
Name: proportion, dtype: float64
****************************************
first_interaction
Website       0.551171
Mobile App    0.448829
Name: proportion, dtype: float64
****************************************
profile_completed
High      0.490893
Medium    0.485906
Low       0.023200
Name: proportion, dtype: float64
****************************************
last_activity
Email Activity      0.493929
Phone Activity      0.267563
Website Activity    0.238508
Name: proportion, dtype: float64
****************************************
print_media_type1
No     0.892238
Yes    0.107762
Name: proportion, dtype: float64
****************************************
print_media_type2
No     0.94948
Yes    0.05052
Name: proportion, dtype: float64
****************************************
digital_media
No     0.885733
Yes    0.114267
Name: proportion, dtype: float64
****************************************
educational_channels
No     0.847138
Yes    0.152862
Name: proportion, dtype: float64
****************************************
referral
No     0.979835
Yes    0.020165
Name: proportion, dtype: float64
****************************************
status
0    0.701431
1    0.298569
Name: proportion, dtype: float64
****************************************
```

In [25]:
```
cols_list=df.select_dtypes(include=np.number).columns.tolist()
plt.figure(figsize=(12,7))
sns.heatmap(df[cols_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectr
plt.show()
```



There are no positive correlations between the previous items

In [26]:
```
#determine how many professionals made sales
df[category_col].query('current_occupation == "Professional" and status==1')
```

Out[26]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
6	Professional	Mobile App	Medium	Website Activity	No	
8	Professional	Mobile App	High	Phone Activity	No	
10	Professional	Website	Medium	Email Activity	No	
11	Professional	Website	High	Website Activity	Yes	
16	Professional	Website	High	Email Activity	No	
...	...	...	...	...	...	
4583	Professional	Website	Medium	Email Activity	Yes	
4587	Professional	Website	High	Email Activity	No	
4588	Professional	Mobile App	Medium	Email Activity	No	
4604	Professional	Website	Medium	Website Activity	No	
4609	Professional	Website	High	Email Activity	No	

929 rows × 10 columns

In [27]:

```
#determine how many Unemployed resulted in sales
df[category_col].query('current_occupation == "Unemployed" and status==1')
```

Out[27]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
0	Unemployed	Website	High	Website Activity	Yes	
3	Unemployed	Website	High	Website Activity	No	
31	Unemployed	Website	Medium	Email Activity	No	
57	Unemployed	Website	Medium	Email Activity	No	
73	Unemployed	Website	High	Phone Activity	No	
...	...	...	...	...	...	
4490	Unemployed	Website	High	Email Activity	No	
4554	Unemployed	Website	High	Email Activity	No	
4555	Unemployed	Website	High	Website Activity	No	
4570	Unemployed	Website	High	Website Activity	Yes	
4574	Unemployed	Website	High	Website Activity	No	

383 rows × 10 columns

In [28]:

```
#determine how many students resulted in sales
df[category_col].query('current_occupation == "Student" and status==1')
```

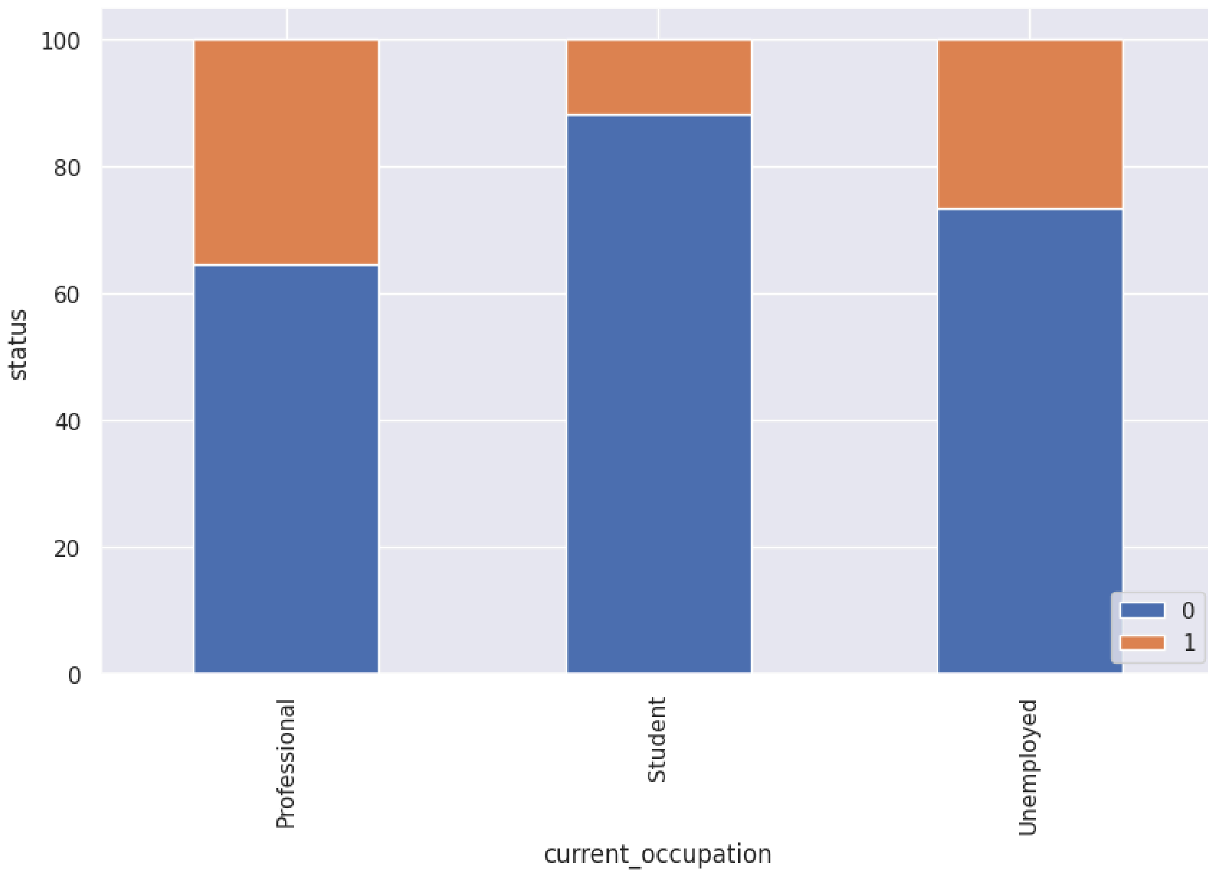
Out[28]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
55	Student	Website	Medium	Website Activity	No	
138	Student	Website	High	Email Activity	No	
207	Student	Website	High	Email Activity	Yes	
299	Student	Mobile App	High	Email Activity	No	
319	Student	Website	Medium	Email Activity	No	
...	...	...	...	...	...	
4345	Student	Website	High	Website Activity	No	
4370	Student	Website	High	Email Activity	No	
4418	Student	Website	High	Website Activity	Yes	
4449	Student	Website	High	Website Activity	No	
4468	Student	Mobile App	Medium	Email Activity	No	

65 rows × 10 columns

In [29]:

```
def stacked_barplot(df, predictor, target, figsize=(10,6)):
  (pd.crosstab(df[predictor], df[target], normalize='index')*100).plot(kind='bar', fig
  plt.legend(loc="lower right")
  plt.ylabel(target)

stacked_barplot(df, "current_occupation", "status")
```



In [30]:
```
#determine how many of each current_occupation there is and how many total interaction
for i in category_col:
  print(df[i].value_counts(normalize=False))
  print('*'*40)
```

```
current_occupation
Professional    2616
Unemployed      1441
Student          555
Name: count, dtype: int64
****************************************
first_interaction
Website       2542
Mobile App    2070
Name: count, dtype: int64
****************************************
profile_completed
High      2264
Medium    2241
Low        107
Name: count, dtype: int64
****************************************
last_activity
Email Activity      2278
Phone Activity      1234
Website Activity    1100
Name: count, dtype: int64
****************************************
print_media_type1
No     4115
Yes     497
Name: count, dtype: int64
****************************************
print_media_type2
No     4379
Yes     233
Name: count, dtype: int64
****************************************
digital_media
No     4085
Yes     527
Name: count, dtype: int64
****************************************
educational_channels
No     3907
Yes     705
Name: count, dtype: int64
****************************************
referral
No     4519
Yes      93
Name: count, dtype: int64
****************************************
status
0    3235
1    1377
Name: count, dtype: int64
****************************************
```

After completing the math, it was found that professionals accounted for 20% of total sales made, unemployed 8% of total sales made, and students only 1% of total sales made. When looking at closing rate though, it was found that we only closed on 12% of professional contacts, 26% of unemployed contacts, and 11% of student contacts. Failure to close on student contacts could be due to a lack of money, but to only be closing on 1% more for professional contacts ;when they have more money and companies investing in their skill growth, is poor. It is interesting that we are closing on 26% of unemployed contacts comparably. This may be because they are using their savings to actively invest in growing or pivoting their skills in order to gain employment. We may want to focus more of our products on assisting these workers and focus on creating more contacts in this space.

In [31]:
```
#Determine how many first interactions via website resulted in sales
df[category_col].query('first_interaction == "Website" and status==1')
```

Out[31]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
0	Unemployed	Website	High	Website Activity	Yes	
3	Unemployed	Website	High	Website Activity	No	
10	Professional	Website	Medium	Email Activity	No	
11	Professional	Website	High	Website Activity	Yes	
16	Professional	Website	High	Email Activity	No	
...	...	...	...	...	...	
4580	Professional	Website	Medium	Email Activity	No	
4583	Professional	Website	Medium	Email Activity	Yes	
4587	Professional	Website	High	Email Activity	No	
4604	Professional	Website	Medium	Website Activity	No	
4609	Professional	Website	High	Email Activity	No	

1159 rows × 10 columns

In [32]:
```
#Determine how many first interactions via mobile app resulted in sales
df[category_col].query('first_interaction == "Mobile App" and status==1')
```

Out[32]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
6	Professional	Mobile App	Medium	Website Activity	No	
8	Professional	Mobile App	High	Phone Activity	No	
24	Professional	Mobile App	High	Email Activity	No	
51	Professional	Mobile App	High	Email Activity	No	
72	Professional	Mobile App	Medium	Email Activity	No	
...	...	...	...	...	...	
4425	Unemployed	Mobile App	High	Website Activity	No	
4455	Professional	Mobile App	Medium	Email Activity	No	
4468	Student	Mobile App	Medium	Email Activity	No	
4528	Professional	Mobile App	Medium	Website Activity	No	
4588	Professional	Mobile App	Medium	Email Activity	No	

218 rows × 10 columns

Once the math is completed, we find that we close on approximately 46% (almost half) of first interactions via the wesite vs only 11% of first interactions via the mobile app. With products and company being the same, there may need to be some explorations into customer complains regarding the mobile app. Maybe a survey of user experience regarding both the website and the mobile app to determine what changes can be made to the app to create a similar experience to using the website.

In [33]:

```
#determine the number of last activity via email resulted in sales
df[category_col].query('last_activity == "Email Activity" and status==1')
```

Out[33]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
10	Professional	Website	Medium	Email Activity	No	
16	Professional	Website	High	Email Activity	No	
24	Professional	Mobile App	High	Email Activity	No	
25	Professional	Website	Medium	Email Activity	No	
31	Unemployed	Website	Medium	Email Activity	No	
...	...	...	...	...	...	
4580	Professional	Website	Medium	Email Activity	No	
4583	Professional	Website	Medium	Email Activity	Yes	
4587	Professional	Website	High	Email Activity	No	
4588	Professional	Mobile App	Medium	Email Activity	No	
4609	Professional	Website	High	Email Activity	No	

691 rows × 10 columns

In [34]:

```
#determine the number of last activity via phone resulted in sales
df[category_col].query('last_activity == "Phone Activity" and status==1')
```

Out[34]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
8	Professional	Mobile App	High	Phone Activity	No	
60	Professional	Website	High	Phone Activity	No	
73	Unemployed	Website	High	Phone Activity	No	
102	Professional	Website	Medium	Phone Activity	No	
132	Unemployed	Website	High	Phone Activity	No	
...	...	...	...	...	...	
4484	Professional	Website	Low	Phone Activity	No	
4485	Professional	Website	High	Phone Activity	No	
4499	Professional	Website	High	Phone Activity	Yes	
4538	Professional	Website	High	Phone Activity	No	
4548	Professional	Website	High	Phone Activity	No	

263 rows × 10 columns

In [35]:

```
#determine the number of last activity via website chat resulted in sales
df[category_col].query('last_activity == "Website Activity" and status==1')
```

Out[35]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
0	Unemployed	Website	High	Website Activity	Yes	
3	Unemployed	Website	High	Website Activity	No	
6	Professional	Mobile App	Medium	Website Activity	No	
11	Professional	Website	High	Website Activity	Yes	
53	Professional	Website	High	Website Activity	No	
...	...	...	...	...	...	
4528	Professional	Mobile App	Medium	Website Activity	No	
4555	Unemployed	Website	High	Website Activity	No	
4570	Unemployed	Website	High	Website Activity	Yes	
4574	Unemployed	Website	High	Website Activity	No	
4604	Professional	Website	Medium	Website Activity	No	

423 rows × 10 columns

Email activity generates the most sales with 691 sales generated. Website activity was second with 423 sales, and phone activity was last with 263 sales.

As far as conversion rates, website activity led with a 38% conversion rate. Email was second at 30%, and phone conversion rate was last at 21%

In [36]:

```
#determine the number of sales generated by print media 1
df[category_col].query('print_media_type1 == "Yes" and status==1')
```

Out[36]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
0	Unemployed	Website	High	Website Activity	Yes	
11	Professional	Website	High	Website Activity	Yes	
56	Professional	Website	High	Website Activity	Yes	
59	Professional	Website	High	Email Activity	Yes	
90	Professional	Website	High	Website Activity	Yes	
...	...	...	...	...	...	
4469	Professional	Website	High	Website Activity	Yes	
4486	Professional	Website	High	Email Activity	Yes	
4499	Professional	Website	High	Phone Activity	Yes	
4570	Unemployed	Website	High	Website Activity	Yes	
4583	Professional	Website	Medium	Email Activity	Yes	

159 rows × 10 columns

In [37]:

```
#Provide conversion rate for print media type 1... total number to divide by recovered
media1conv=len(df[category_col].query('print_media_type1 == "Yes" and status==1'))
print(media1conv/497)
```

0.3199195171026157

In [38]:

```
#determine the number of sales generated by print media 2
df[category_col].query('print_media_type2 == "Yes" and status==1')
```

Out[38]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
11	Professional	Website	High	Website Activity	Yes	
76	Professional	Website	High	Website Activity	No	
114	Professional	Website	High	Email Activity	No	
139	Professional	Website	High	Email Activity	No	
284	Professional	Website	High	Email Activity	No	
...	...	...	...	...	...	
4293	Professional	Website	Medium	Website Activity	No	
4316	Professional	Mobile App	Medium	Website Activity	No	
4357	Professional	Website	Medium	Email Activity	No	
4466	Professional	Website	Medium	Email Activity	No	
4506	Professional	Website	High	Email Activity	No	

75 rows × 10 columns

In [39]:
```
#Provide conversion rate for print media type 2... total number to divide by recovered
media2conv=len(df[category_col].query('print_media_type2 == "Yes" and status==1'))
print(media2conv/233)
```

```
0.3218884120171674
```

In [40]:
```
#determine the number of sales generated by digital media
df[category_col].query('digital_media == "Yes" and status==1')
```

Out[40]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
0	Unemployed	Website	High	Website Activity	Yes	
6	Professional	Mobile App	Medium	Website Activity	No	
8	Professional	Mobile App	High	Phone Activity	No	
16	Professional	Website	High	Email Activity	No	
31	Unemployed	Website	Medium	Email Activity	No	
...	...	...	...	...	...	
4369	Unemployed	Website	High	Email Activity	No	
4411	Professional	Website	Medium	Email Activity	No	
4421	Professional	Website	Medium	Website Activity	No	
4423	Professional	Website	High	Email Activity	No	
4445	Unemployed	Website	High	Email Activity	No	

168 rows × 10 columns

In [41]:

```
#Provide conversion rate for digital media... total number to divide by recovered from
digitalconv=len(df[category_col].query('digital_media == "Yes" and status==1'))
print(digitalconv/527)
```

```
0.3187855787476281
```

In [42]:

```
#determine the number of sales generated by educational channels
df[category_col].query('educational_channels == "Yes" and status==1')
```

Out[42]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
25	Professional	Website	Medium	Email Activity	No	
31	Unemployed	Website	Medium	Email Activity	No	
53	Professional	Website	High	Website Activity	No	
55	Student	Website	Medium	Website Activity	No	
56	Professional	Website	High	Website Activity	Yes	
...	...	...	...	...	...	
4407	Professional	Website	High	Email Activity	No	
4417	Unemployed	Website	Medium	Website Activity	No	
4420	Unemployed	Website	High	Email Activity	No	
4449	Student	Website	High	Website Activity	No	
4550	Professional	Website	Medium	Email Activity	No	

197 rows × 10 columns

In [43]:

```
#Provide conversion rate for educational channels... total number to divide by recover
educonv=len(df[category_col].query('educational_channels == "Yes" and status==1'))
print(educonv/705)
```

```
0.2794326241134752
```

In [44]:

```
#determine the number of sales generated by referrals
df[category_col].query('referral == "Yes" and status==1')
```

Out[44]:

	current_occupation	first_interaction	profile_completed	last_activity	print_media_type1	print_m
16	Professional	Website	High	Email Activity	No	
52	Professional	Website	High	Email Activity	No	
253	Professional	Mobile App	Medium	Website Activity	No	
306	Professional	Website	High	Website Activity	Yes	
388	Professional	Mobile App	High	Phone Activity	Yes	
...	...	...	...	...	...	
4368	Professional	Website	High	Email Activity	No	
4370	Student	Website	High	Email Activity	No	
4468	Student	Mobile App	Medium	Email Activity	No	
4476	Professional	Website	High	Email Activity	No	
4550	Professional	Website	Medium	Email Activity	No	

63 rows × 10 columns

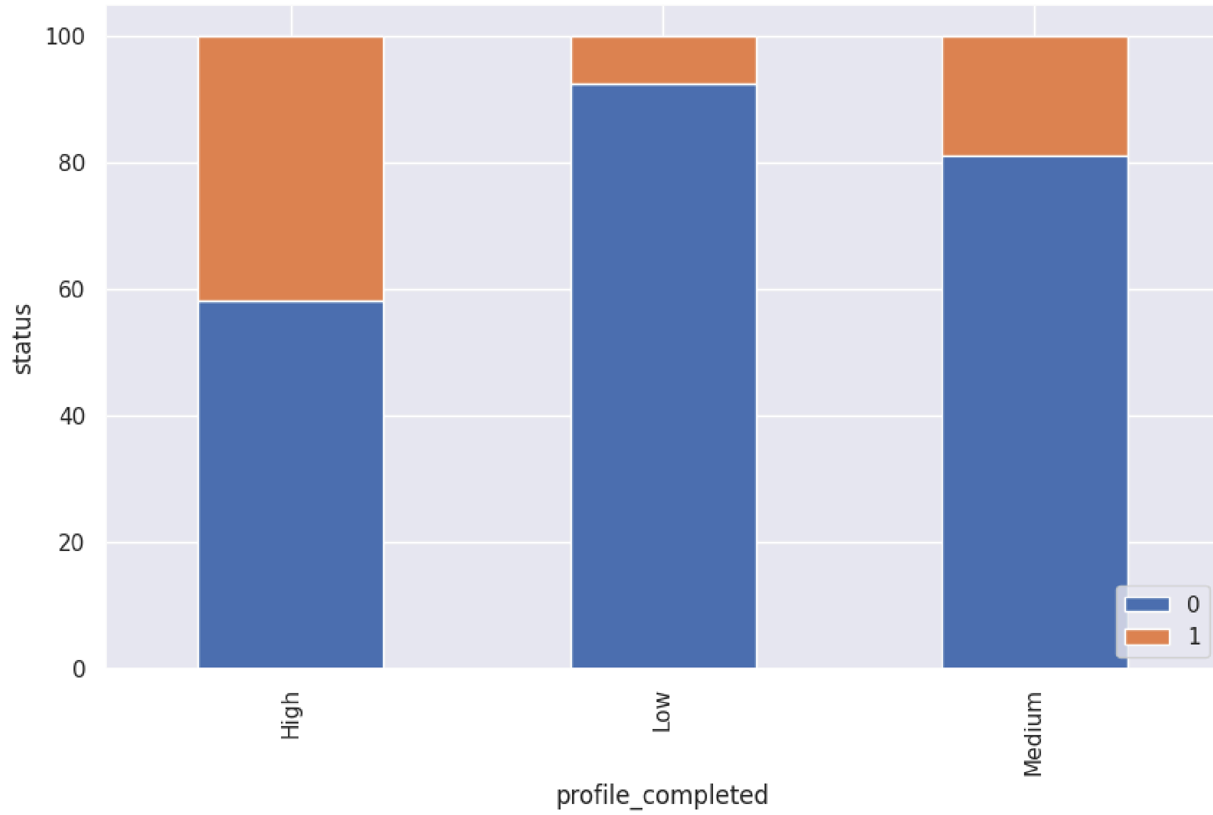
In [45]:

```
#Provide conversion rate for referrals... total number to divide by recovered from abc
referralconv=len(df[category_col].query('referral == "Yes" and status==1'))
print(referralconv/93)
```

```
0.6774193548387096
```

In [46]:

```
### Defining the stacked_barplot() function and running comparison on profile complete
def stacked_barplot(data,predictor,target,figsize=(10,6)):
  (pd.crosstab(data[predictor],data[target],normalize='index')*100).plot(kind='bar',fi
  plt.legend(loc="lower right")
  plt.ylabel(target)

stacked_barplot(df, "profile_completed", "status")
```



In [47]:
```
df.columns  ## checking data
```

Out[47]:
```
Index(['age', 'current_occupation', 'first_interaction', 'profile_completed',
       'website_visits', 'time_spent_on_website', 'page_views_per_visit',
       'last_activity', 'print_media_type1', 'print_media_type2',
       'digital_media', 'educational_channels', 'referral', 'status'],
      dtype='object')
```

In [48]:
```
df.info()   ### checking data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4612 entries, 0 to 4611
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype   
---  ------                 --------------  -----   
 0   age                    4612 non-null   int64   
 1   current_occupation     4612 non-null   category
 2   first_interaction      4612 non-null   category
 3   profile_completed      4612 non-null   category
 4   website_visits         4612 non-null   int64   
 5   time_spent_on_website  4612 non-null   int64   
 6   page_views_per_visit   4612 non-null   float64 
 7   last_activity          4612 non-null   category
 8   print_media_type1      4612 non-null   category
 9   print_media_type2      4612 non-null   category
 10  digital_media          4612 non-null   category
 11  educational_channels   4612 non-null   category
 12  referral               4612 non-null   category
 13  status                 4612 non-null   category
dtypes: category(10), float64(1), int64(3)
memory usage: 190.5 KB
```

In [56]:
```
for column in category_col:
  df[column]=df[column].astype('category')
```

```
### converting dtype to category and checking data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4612 entries, 0 to 4611
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype   
---  ------                 --------------  -----   
 0   age                    4612 non-null   int64   
 1   current_occupation     4612 non-null   category
 2   first_interaction      4612 non-null   category
 3   profile_completed      4612 non-null   category
 4   website_visits         4612 non-null   int64   
 5   time_spent_on_website  4612 non-null   int64   
 6   page_views_per_visit   4612 non-null   float64 
 7   last_activity          4612 non-null   category
 8   print_media_type1      4612 non-null   category
 9   print_media_type2      4612 non-null   category
 10  digital_media          4612 non-null   category
 11  educational_channels   4612 non-null   category
 12  referral               4612 non-null   category
 13  status                 4612 non-null   category
dtypes: category(10), float64(1), int64(3)
memory usage: 190.5 KB
```

In [53]:
```
df.info()   ### checking data structure
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4612 entries, 0 to 4611
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype   
---  ------                 --------------  -----   
 0   age                    4612 non-null   int64   
 1   current_occupation     4612 non-null   category
 2   first_interaction      4612 non-null   category
 3   profile_completed      4612 non-null   category
 4   website_visits         4612 non-null   int64   
 5   time_spent_on_website  4612 non-null   int64   
 6   page_views_per_visit   4612 non-null   float64 
 7   last_activity          4612 non-null   category
 8   print_media_type1      4612 non-null   category
 9   print_media_type2      4612 non-null   category
 10  digital_media          4612 non-null   category
 11  educational_channels   4612 non-null   category
 12  referral               4612 non-null   category
 13  status                 4612 non-null   category
dtypes: category(10), float64(1), int64(3)
memory usage: 190.5 KB
```

In [59]:
```
df ### checking how data is now constructed
```

Out[59]:

	age	current_occupation	first_interaction	profile_completed	website_visits	time_spent_on_websi
0	57	Unemployed	Website	High	7	16
1	56	Professional	Mobile App	Medium	2	
2	52	Professional	Website	Medium	3	3
3	53	Unemployed	Website	High	4	4
4	23	Student	Website	High	4	6
...	...	...	...	...	...	
4607	35	Unemployed	Mobile App	Medium	15	3
4608	55	Professional	Mobile App	Medium	8	23
4609	58	Professional	Website	High	2	2
4610	57	Professional	Mobile App	Medium	1	1
4611	55	Professional	Website	Medium	4	22

4612 rows × 14 columns

In [57]:

```
### Creating metric function
def metrics_score(actual, predicted):
    print(classification_report(actual, predicted))

    cm=confusion_matrix(actual, predicted)
    plt.figure(figsize=(8,5))

    sns.heatmap(cm, annot=True, fmt='2f', xticklabels=['Not Converted', 'Converted'],
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()
```

Building a Decision Tree model

In [62]:

```
### it is most important to properly predict the conversion.  It will be unacceptable
### For this, we want Precision to be maximized

#splitting data into 70% train and 30% test sets
X_train, X_test, y_train, y_test= train_test_split(X, Y, test_size=0.3, random_state=1

####  create dummy list
to_get_dummies_for = ['current_occupation', 'first_interaction', 'profile_completed',
```

```
df1=pd.DataFrame(df.drop(columns=['age', 'website_visits', 'time_spent_on_website', 'p

#### creating dummy variables
df1=pd.get_dummies(data=df1, columns=to_get_dummies_for, drop_first=True)

###Mapping referral and status
dict_referral={'1': 1, '0' : 0}
dict_status= {'1': 1, '0' : 0}

###Separating the independent variables (x) and the dependent variable (Y)
Y=df1.status_1
X=df1.drop(columns=['status_1'])


#scaling the data
sc=StandardScaler()

for column in X_train.columns:
    if X_train[column].dtype == "object":
        print(f"Column '{column}' contains non-numeric values.")

#Fit_transform on train data
sc=StandardScaler()
X_train_scaled=sc.fit_transform(X_train)
X_train_scaled=pd.DataFrame(X_train_scaled, columns=X.columns)

##Transform on test data
x_test_scaled=sc.transform(X_test)
x_test_scaled=pd.DataFrame(x_test_scaled, columns=X.columns)

##Building Decision Tree Model
dt=DecisionTreeClassifier(class_weight = {0:0.17, 1:0.83}, random_state=1)

###Fitting decision tree model
dt.fit(X_train, y_train)
```

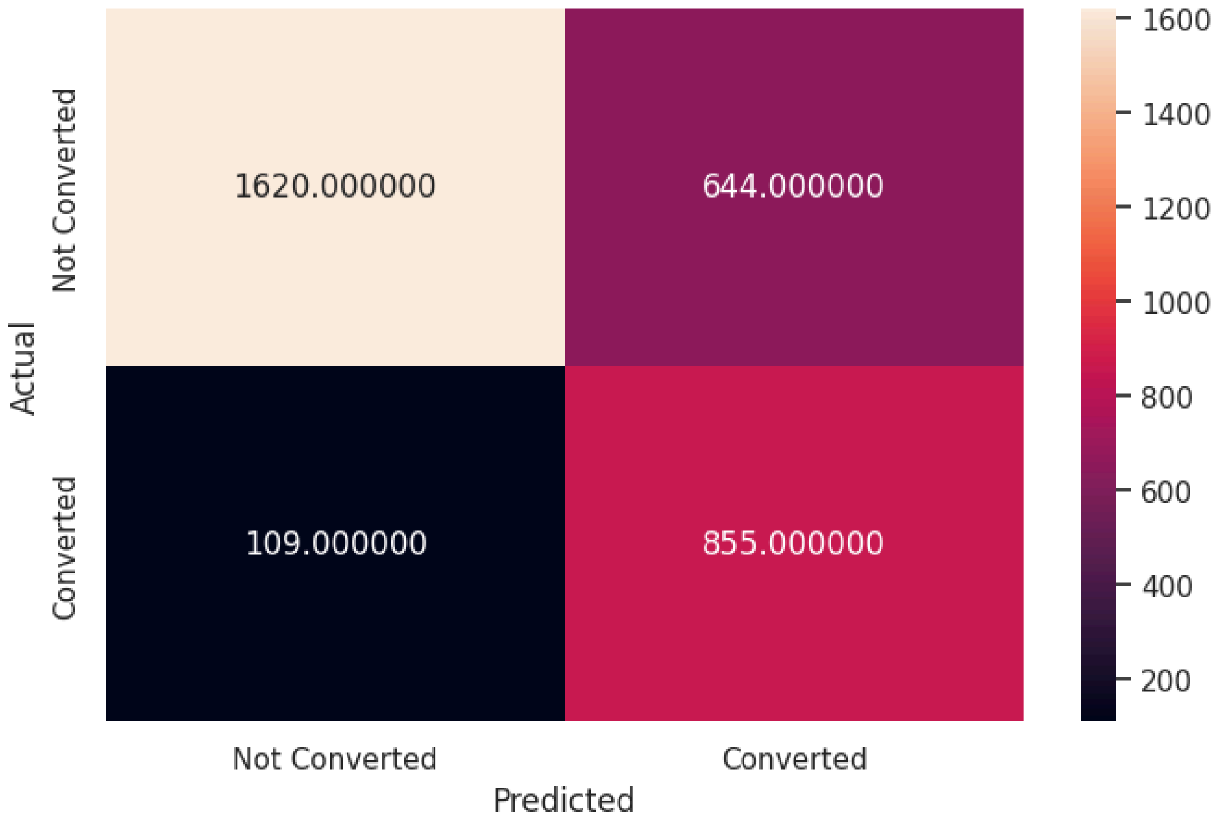
Out[62]:

```
DecisionTreeClassifier
DecisionTreeClassifier(class_weight={0: 0.17, 1: 0.83}, random_state=1)
```

In [64]:

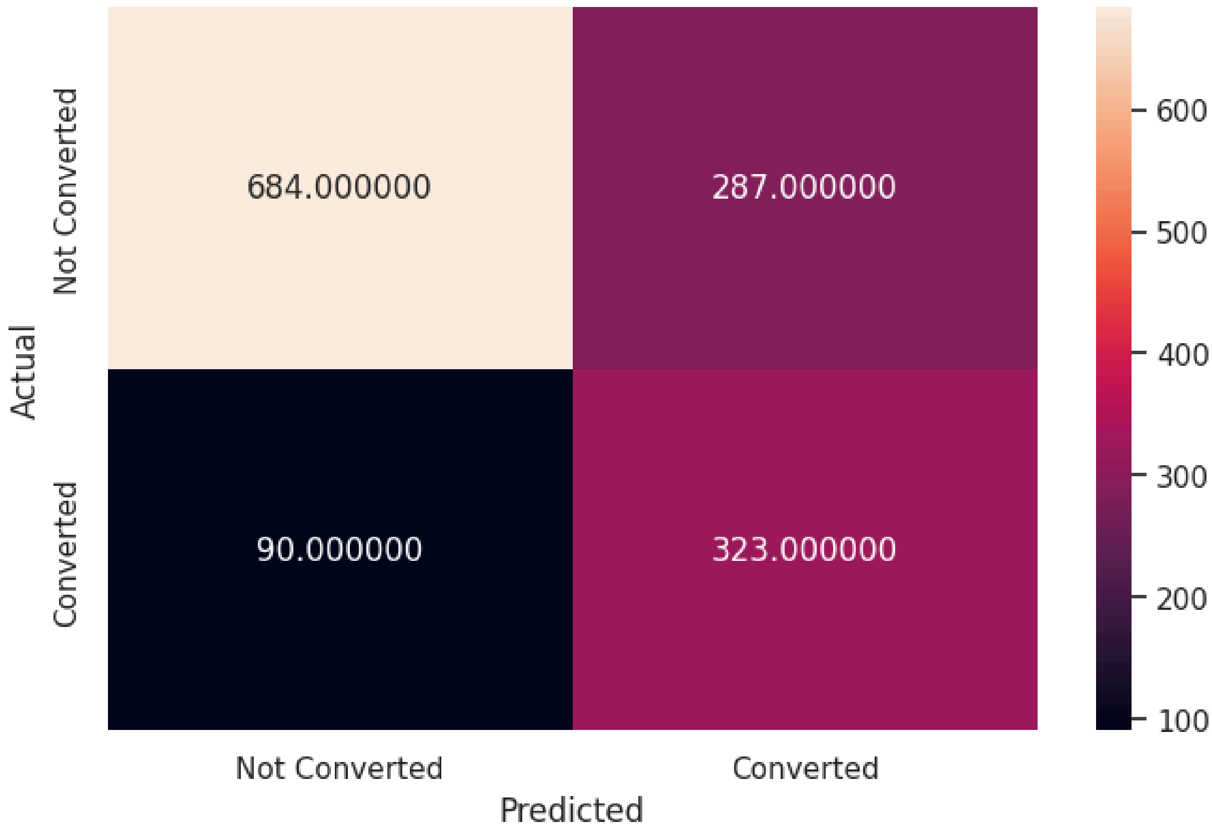
```
### checking performance on the training dataset
y_train_pred_dt=dt.predict(X_train)
metrics_score(y_train, y_train_pred_dt)
```

```
              precision    recall  f1-score   support

       False       0.94      0.72      0.81      2264
        True       0.57      0.89      0.69       964

    accuracy                           0.77      3228
   macro avg       0.75      0.80      0.75      3228
weighted avg       0.83      0.77      0.78      3228
```



In [65]:
```
#### checking performance on test data set
y_test_pred_dt=dt.predict(X_test)
metrics_score(y_test, y_test_pred_dt)
```

```
              precision    recall  f1-score   support

       False       0.88      0.70      0.78       971
        True       0.53      0.78      0.63       413

    accuracy                           0.73      1384
   macro avg       0.71      0.74      0.71      1384
weighted avg       0.78      0.73      0.74      1384
```

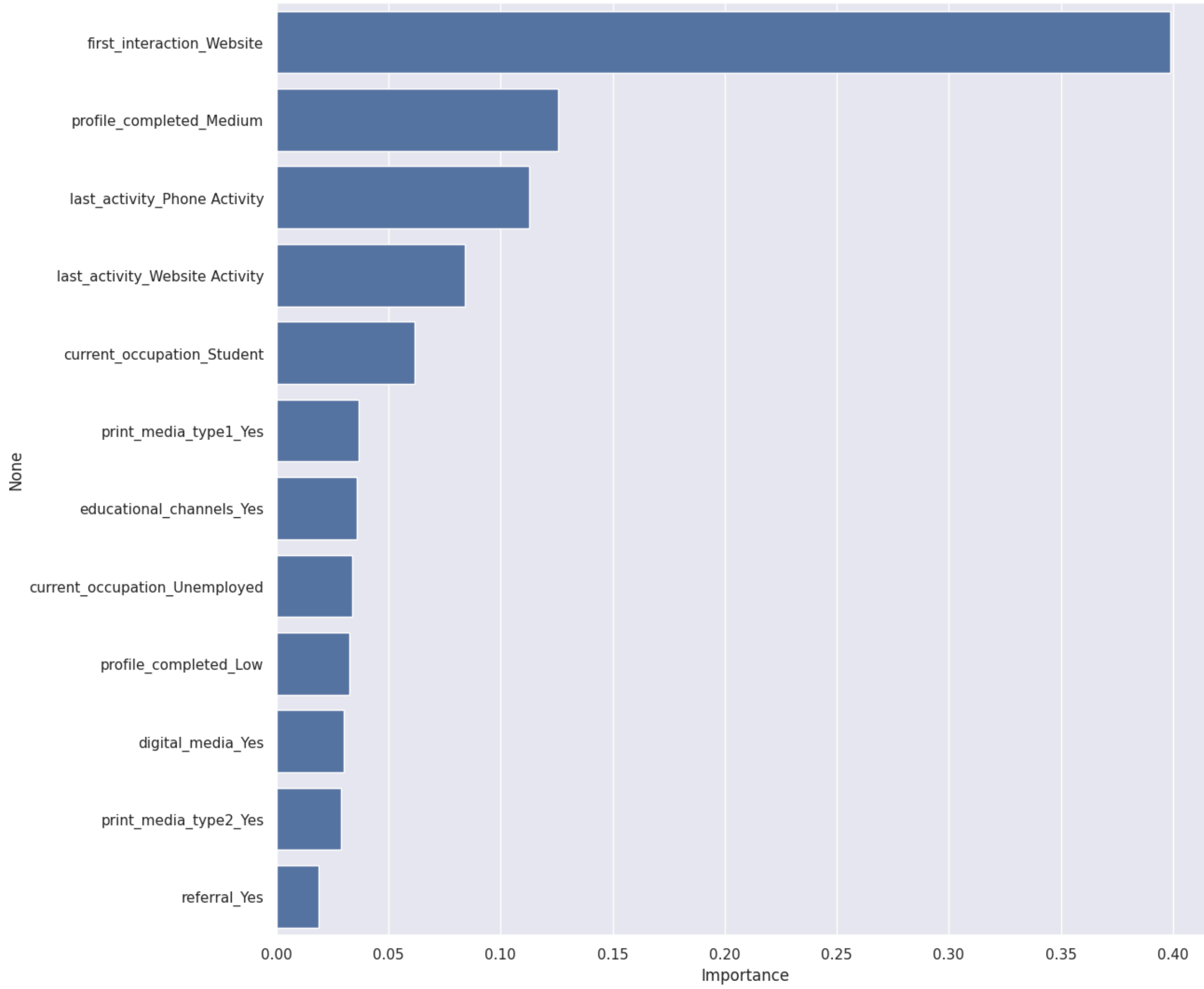


In [68]:
```
confusion_matrix(y_test, y_test_pred_dt)   #another way of doing confusion matrix
```

Out[68]:
```
array([[684, 287],
       [ 90, 323]])
```

In [66]:
```
### plot feature importance
importances = dt.feature_importances_

columns = X.columns

importance_df = pd.DataFrame(importances, index = columns, columns = ['Importance']).s

plt.figure(figsize = (13, 13))

sns.barplot(x=importance_df.Importance, y=importance_df.index)
```

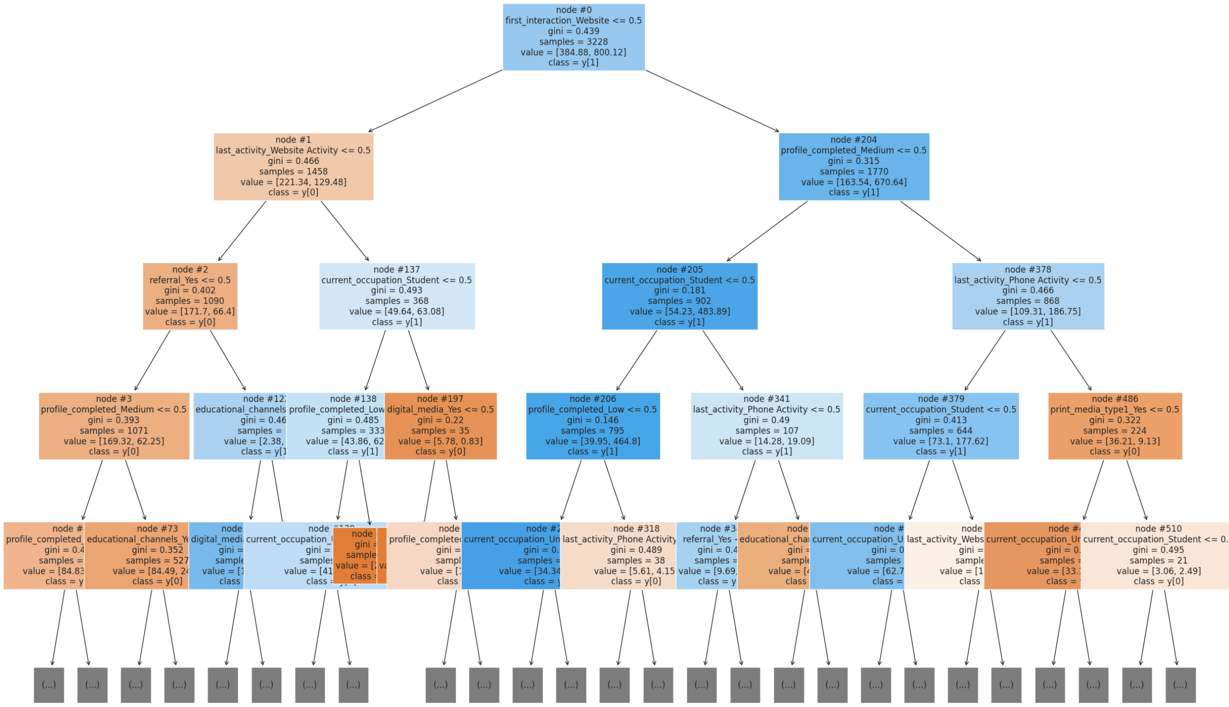
Out[66]:
```
<Axes: xlabel='Importance', ylabel='None'>
```



Looking at this chart, we definitely have overfitting, as many of the features after channels have below a .05 importance

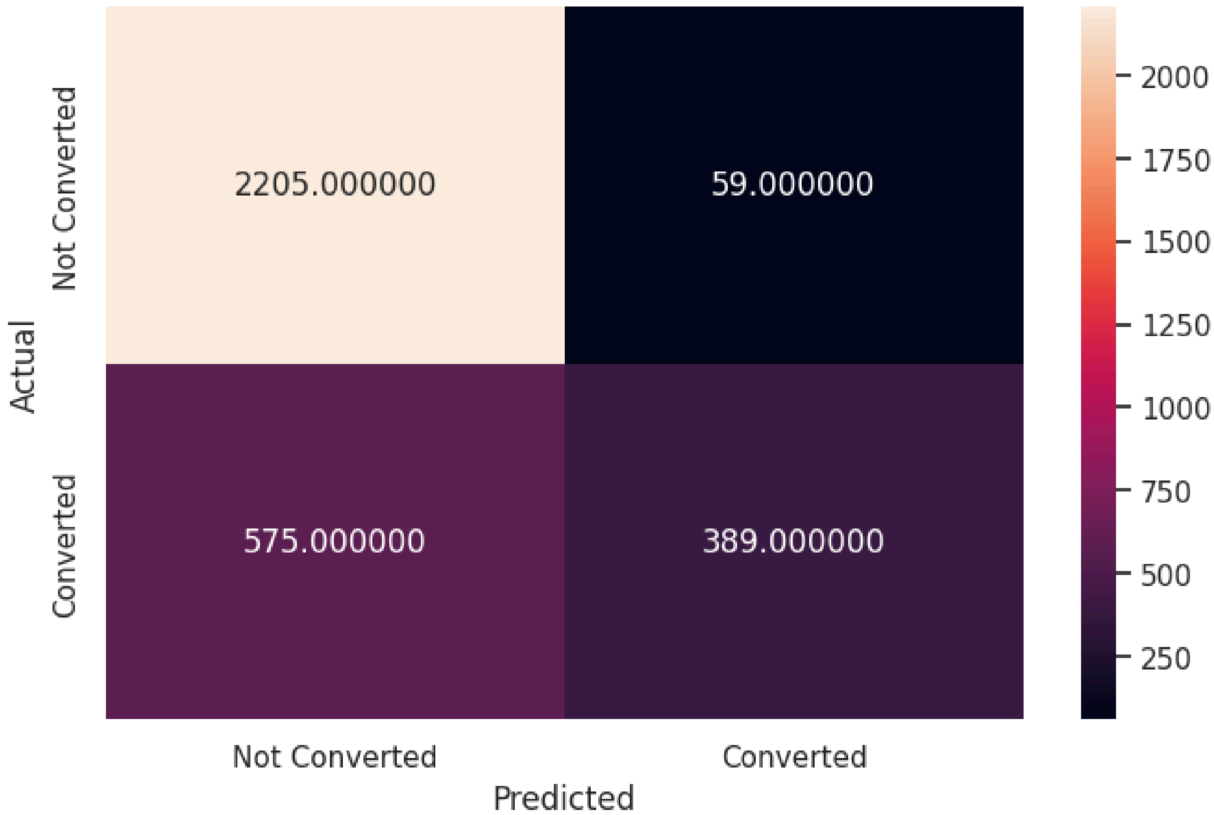
Do we need to prune the tree?

In [69]:
```
#My recall is good, but my precision is low at .53 and .57 for my training, and test d
### We need to prune the tree
## displaying decision tree
features=list(X.columns)
plt.figure(figsize=(30,20))
tree.plot_tree(dt, max_depth = 4, feature_names=features, filled=True, fontsize=12, nc
plt.show()
```




```
##Building New Decision Tree Model
dt=DecisionTreeClassifier(class_weight = {0:0.15, 1:0.05}, random_state=1)

###Fitting decision tree model
dt.fit(X_train, y_train)

### checking performance on the training dataset
y_train_pred_dt=dt.predict(X_train)
metrics_score(y_train, y_train_pred_dt)
```

```
              precision    recall  f1-score   support

       False       0.79      0.97      0.87      2264
        True       0.87      0.40      0.55       964

    accuracy                           0.80      3228
   macro avg       0.83      0.69      0.71      3228
weighted avg       0.82      0.80      0.78      3228
```

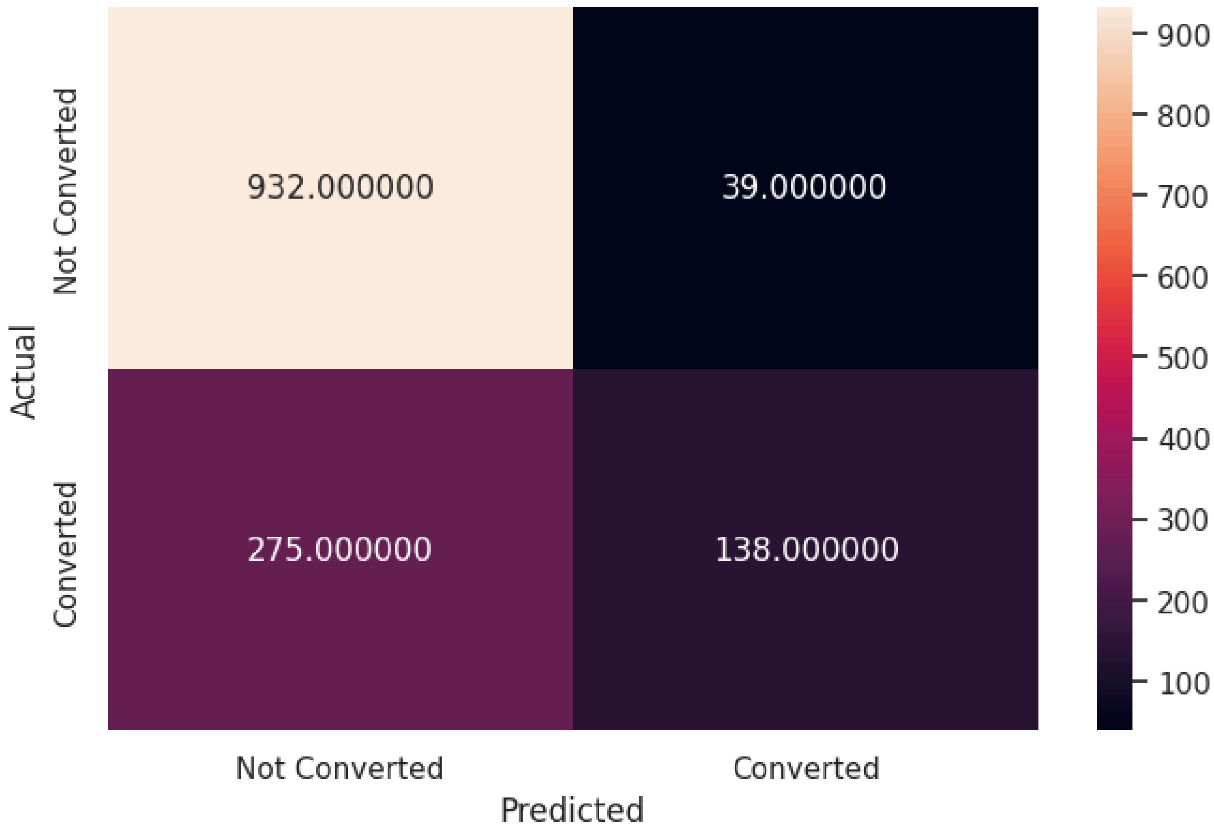


By changing my weights to 0.17 and 0.20, I was able to improve my precision for positives to 0.87 at the expense of my recall coming down to .40 for positives. I am accepting precision up as high as possible as false positives are not acceptable and false negatives are acceptable; causing poor recall, but doesn't not affect projecttions of company goals negatively. I would like to have a better recall in the long term so that I can project necessary resources to have available to provide services to the customer.

In [71]:

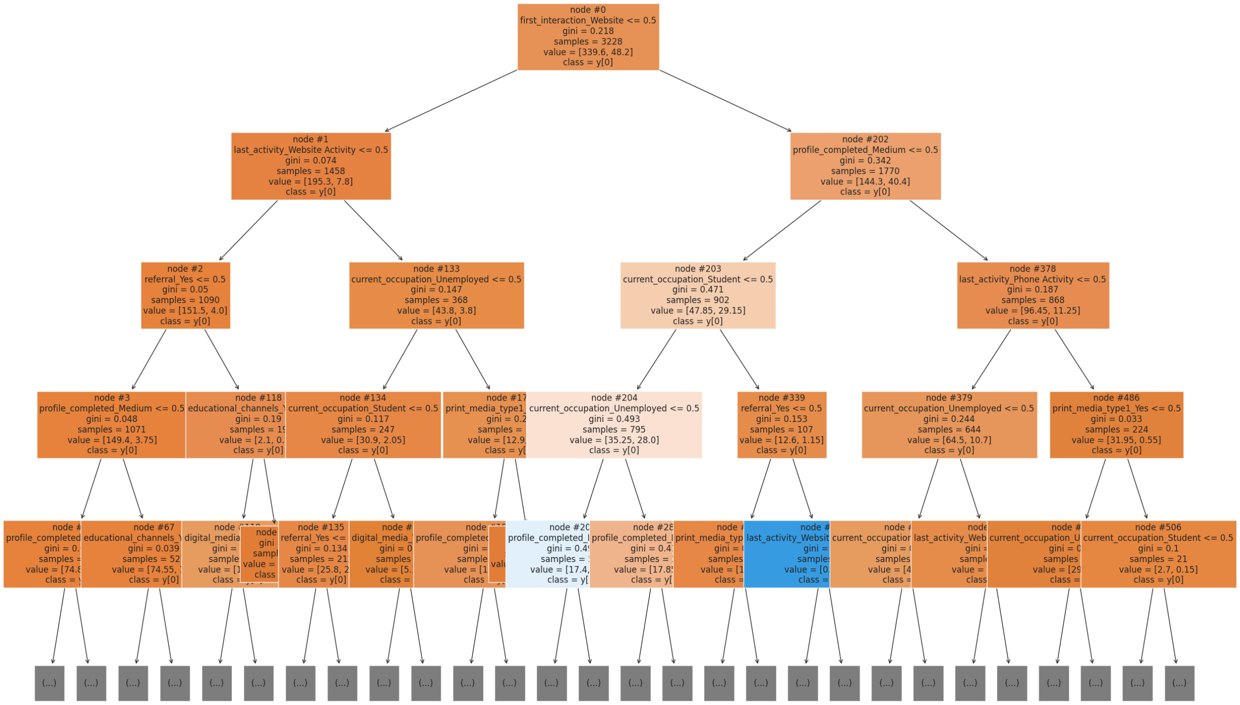
```
### checking performance on test data set
y_test_pred_dt=dt.predict(X_test)
metrics_score(y_test, y_test_pred_dt)
```

```
              precision    recall  f1-score   support

       False       0.77      0.96      0.86       971
        True       0.78      0.33      0.47       413

    accuracy                           0.77      1384
   macro avg       0.78      0.65      0.66      1384
weighted avg       0.77      0.77      0.74      1384
```



1) We definitely need to prune our data as with training, our precision dropped to .78 with a recall of .33 looking at the test data

In [72]:
```
features=list(X.columns)
plt.figure(figsize=(30,20))
tree.plot_tree(dt, max_depth = 4, feature_names=features, filled=True, fontsize=12, nc
plt.show()
```



In [81]:

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid to search over
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the decision tree classifier
dt = DecisionTreeClassifier()

# Create the GridSearchCV object
grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=5, scoring='accurac
# 'accuracy' can be replaced by other metrics like 'precision', 'recall', 'f1' etc.

# Fit the grid search to your training data
grid_search.fit(X_train, y_train) # Assuming you have X_train and y_train defined

# Print the best parameters found
print("Best parameters: ", grid_search.best_params_)
```

```
Best parameters:  {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'mi
n_samples_split': 2}
```

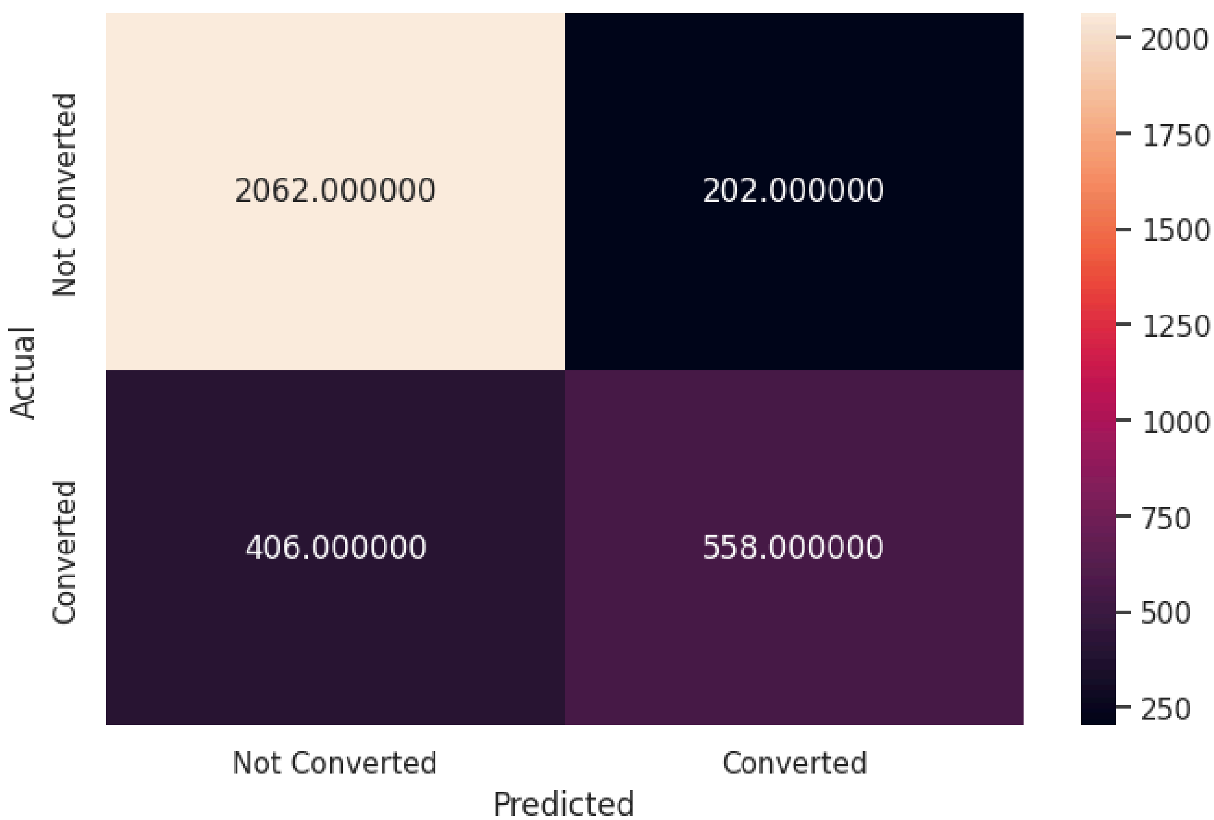
In [82]:

```
# Assuming 'grid_search' is the GridSearchCV object from the previous code
best_params = grid_search.best_params_

# Create a new DecisionTreeClassifier with the best parameters
best_dt_model = DecisionTreeClassifier(**best_params)

# Fit the model to your training data
best_dt_model.fit(X_train, y_train)
```

```python
# Now you can use best_dt_model to make predictions
y_pred = best_dt_model.predict(X_test)  # Assuming you have X_test defined
```

In [85]:
```python
###Training last model


### checking performance on the training dataset
y_pred=best_dt_model.predict(X_train)
metrics_score(y_train, y_pred)
```

```
              precision    recall  f1-score   support

       False       0.84      0.91      0.87      2264
        True       0.73      0.58      0.65       964

    accuracy                           0.81      3228
   macro avg       0.78      0.74      0.76      3228
weighted avg       0.81      0.81      0.80      3228
```

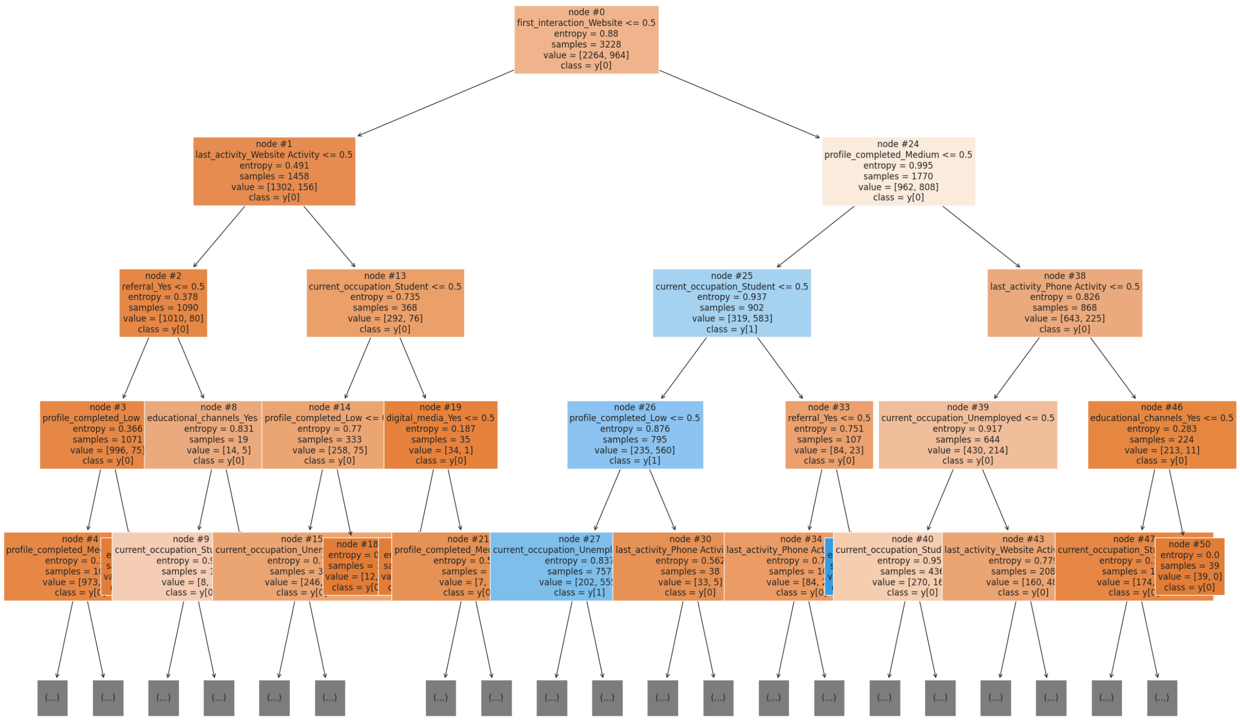


In [86]:
```python
#Final Test on Test Data
### checking performance on test data set
y_pred=best_dt_model.predict(X_test)
metrics_score(y_test, y_pred)
```

```
              precision    recall  f1-score   support

       False       0.84      0.90      0.87       971
        True       0.72      0.58      0.64       413

    accuracy                           0.81      1384
   macro avg       0.78      0.74      0.75      1384
weighted avg       0.80      0.81      0.80      1384
```

After the Grid search CV and applicaton, we sacrificed some precision but improved slightly on our recall and F1 score. We have a very high recall and F1 score on our predictions of non converted sales, so that should help us cross check and make up for some of our loss of precision False positives. Ultimately, this implies that we will have a lot of false predictions of unclosed sales in our model, so we may want to prepare to have some more resources on site to provide services if we use this model.

In [87]:
```
features=list(X.columns)
plt.figure(figsize=(30,20))
tree.plot_tree(best_dt_model, max_depth = 4, feature_names=features, filled=True, font
plt.show()
```



Building a Random Forest model

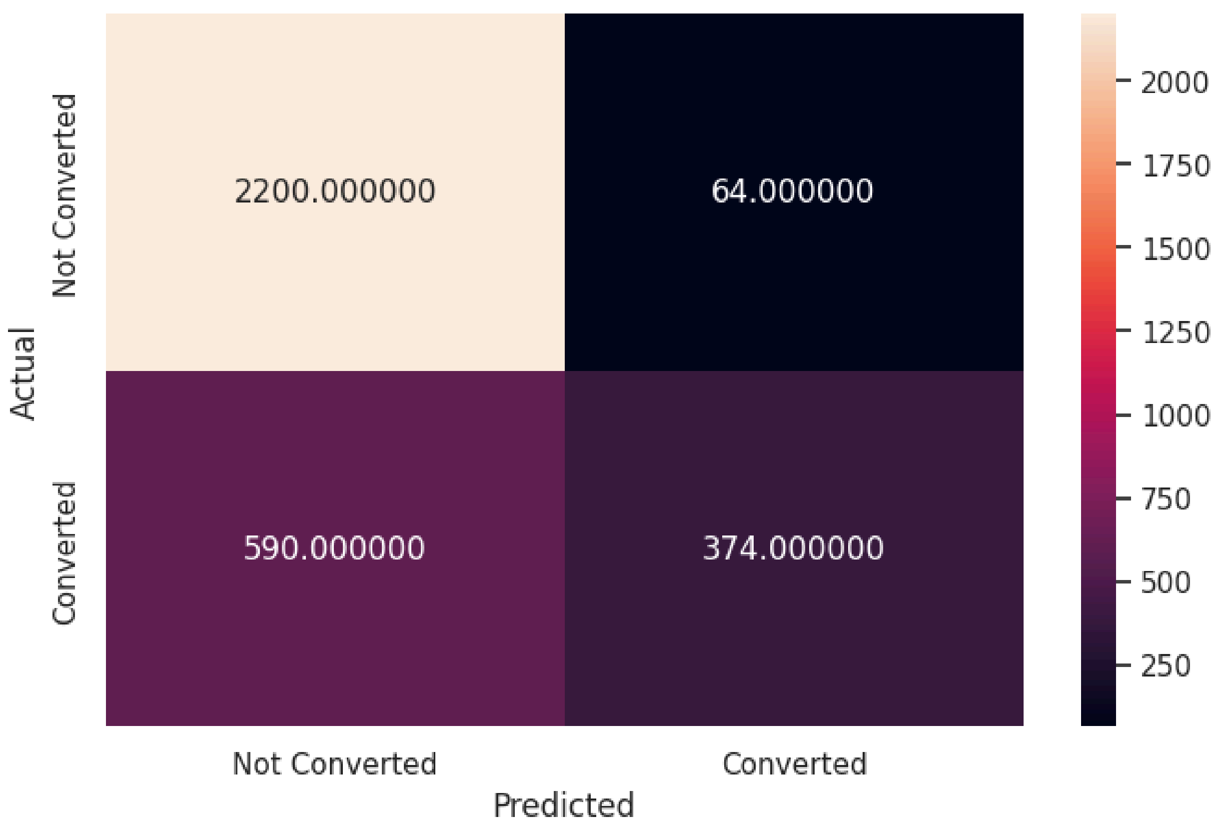
In [73]:
```python
### fitting the random forest classifier on the training data
rf_estimator_tuned=RandomForestClassifier(class_weight = {0: 0.15, 1: 0.05}, random_st

# Grid of parameters to choose from
params_rf = {
        "n_estimators": [100, 250, 500],
        "min_samples_leaf": np.arange(1, 4, 1),
        "max_features": [0.7, 0.9, 'auto'],}



# Run the grid search
grid_obj = GridSearchCV(rf_estimator_tuned, params_rf,  cv = 5)

grid_obj = grid_obj.fit(X_train, y_train)



# Set the classifier to the best combination of parameters
rf_estimator_tuned = grid_obj.best_estimator_
```

In [74]:
```python
rf_estimator_tuned.fit(X_train, y_train)
```

Out[74]:
```
▾ RandomForestClassifier
RandomForestClassifier(class_weight={0: 0.15, 1: 0.05}, max_features=0.7,
                       min_samples_leaf=2, n_estimators=250, random_state=1)
```

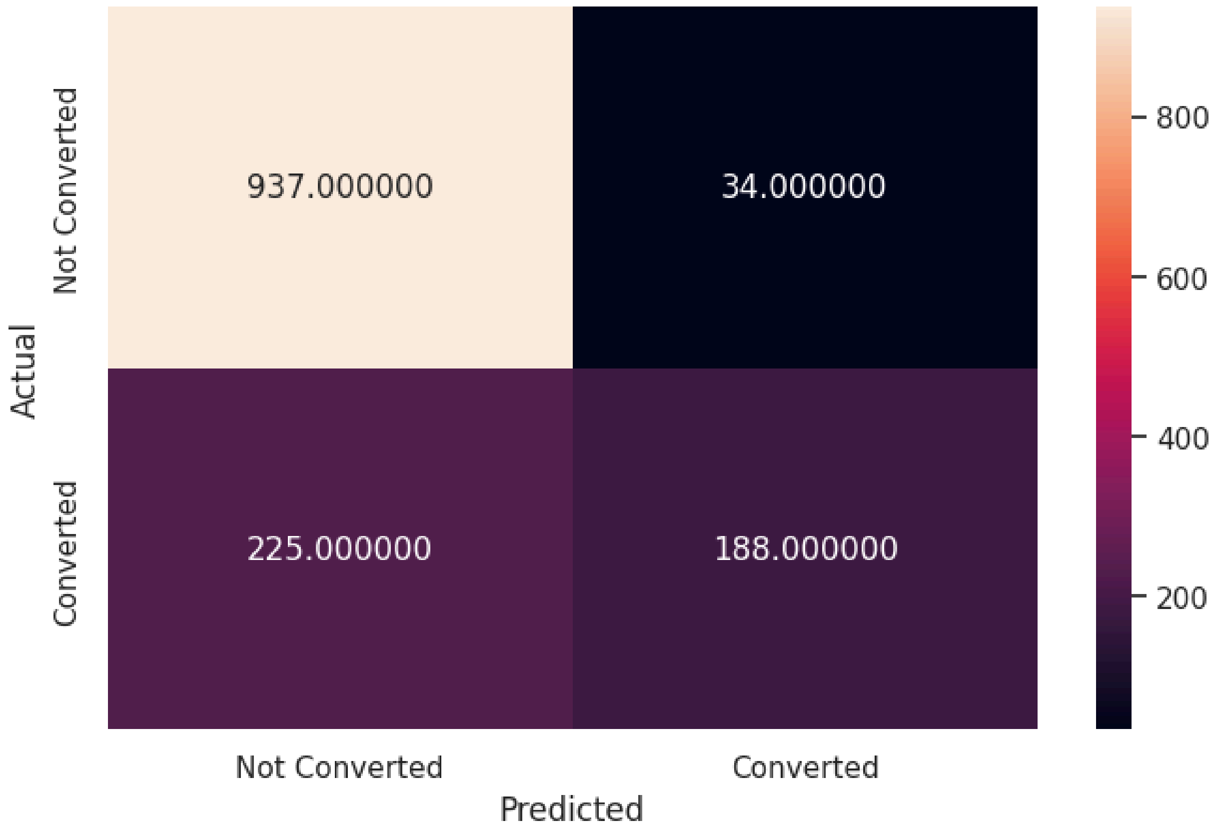
In [77]:
```
## checking performancce on the training data
# Fit the RandomForestClassifier
rf_estimator_tuned.fit(X_train, y_train)

# Predict on the training data
y_pred_train_rf = rf_estimator_tuned.predict(X_train)

# Evaluate the performance
metrics_score(y_train, y_pred_train_rf)
```

```
              precision    recall  f1-score   support

       False       0.79      0.97      0.87      2264
        True       0.85      0.39      0.53       964

    accuracy                           0.80      3228
   macro avg       0.82      0.68      0.70      3228
weighted avg       0.81      0.80      0.77      3228
```



In [78]:
```
## checking performancce on the test data
# Fit the RandomForestClassifier
rf_estimator_tuned.fit(X_test, y_test)

# Predict on the training data
y_pred_test_rf = rf_estimator_tuned.predict(X_test)

# Evaluate the performance
metrics_score(y_test, y_pred_test_rf)
```

```
              precision    recall  f1-score   support

       False       0.81      0.96      0.88       971
        True       0.85      0.46      0.59       413

    accuracy                           0.81      1384
   macro avg       0.83      0.71      0.74      1384
weighted avg       0.82      0.81      0.79      1384
```



Random forest definitely improved our numbers in a positive way as once compared to test data precision remained at .85 for positive sale conversions. It did verify acceptance of the lower recall, but recall and precision are both strong on negative sales conversions, so that data can be referenced for further insite on false negatives.

Do we need to prune the tree?

In [ ]:

```
###The Random forest Tree does not need to be pruned
```

Actionable Insights and Recommendations

In [79]:

```
###1.  Focus more efforts into creating products for unemployed individuals and creati

###2.  Begin research into why the mobile app is performing so poorly compared to the
      ###  appears that there may simply be some customer dissatisfaction with mobile
```

```
###3.  Work on generating more referral leads.  Possibly with redirected funds from pr

###4.  Referrals, profile completeness, first interactions, and current occupation app
      ###  profile completeness, last activity, and lead channel have a lesser but s
```

1. Focus more efforts into creating products for unemployed individuals and creating contacts in that demographic

2. Begin research into why the mobile app is performing so poorly compared to the Website. Appears that there may simply be some customer dissatisfaction with mobile apps build or performance.

3. Work on generating more referral leads. Possibly with redirected funds from print media 2 due to an extreme lack of generated lead volume.

4. Referrals, profile completeness, first interactions, and current occupation appear to be the greatest determiners of sales conversions. Profile completeness, last activity, and lead channel have a lesser but still important impact as well.

5. Based on the work done here, the decision tree model would be the best model to deploy.