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 regularly, 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 interact with ExtraaLearn. Values include 'Website', 'Mobile App'
- profile_completed: What percentage of the 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 the program through email, Representative shared information with a lead like a brochure of program, etc
    - Phone Activity: Had a Phone Conversation with a representative, Had conversation over SMS with a representative, etc
    - Website Activity: Interacted on live chat with a representative, Updated profile on the 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.

Please read the instructions carefully before starting the project.

This is a commented Jupyter IPython Notebook file in which all the instructions and tasks to be performed are mentioned.

- Blanks '_______' are provided in the notebook that

needs to be filled with an appropriate code to get the correct result. With every '_______' blank, there is a comment that briefly describes what needs to be filled in the blank space.

- Identify the task to be performed correctly, and only then proceed to write the required code.
- Fill the code wherever asked by the commented lines like "# write your code here" or "# complete the code". Running incomplete code may throw error.
- Please run the codes in a sequential manner from the beginning to avoid any unnecessary errors.
- Add the results/observations (wherever mentioned) derived from the analysis in the presentation and submit the same.

In [3]:
```
import warnings

warnings.filterwarnings("ignore")
from statsmodels.tools.sm_exceptions import ConvergenceWarning

warnings.simplefilter("ignore", ConvergenceWarning)

# Libraries to help with reading and manipulating data

import pandas as pd
import numpy as np

# Library to split data
from sklearn.model_selection import train_test_split

# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)
# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)
# setting the precision of floating numbers to 5 decimal points
```

```
pd.set_option("display.float_format", lambda x: "%.5f" % x)

# To build model for prediction
import statsmodels.stats.api as sms
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
from statsmodels.tools.tools import add_constant
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier

# To tune different models
from sklearn.model_selection import GridSearchCV


# To get diferent metric scores
import sklearn.metrics as metrics
from sklearn.metrics import (
    f1_score,
    accuracy_score,
    recall_score,
    precision_score,
    confusion_matrix,
    classification_report,
    roc_auc_score,
    precision_recall_curve,
    roc_curve,
    make_scorer,
)
```

Import Dataset

In [5]:
```
learn = pd.read_csv("ExtraaLearn.csv") ##  Complete the code to read the dat
```

In [6]:
```
# copying data to another variable to avoid any changes to original data
data = learn.copy()
```

View the first and last 5 rows of the dataset

In [8]:
```
data.head() ##  Complete the code to view top 5 rows of the data
```

Out[8]:

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

In [9]:
```
data.tail() ##  Complete the code to view last 5 rows of the data
```

Out[9]:

	ID	age	current_occupation	first_interaction	profile_completed	website
4607	EXT4608	35	Unemployed	Mobile App	Medium	
4608	EXT4609	55	Professional	Mobile App	Medium	
4609	EXT4610	58	Professional	Website	High	
4610	EXT4611	57	Professional	Mobile App	Medium	
4611	EXT4612	55	Professional	Website	Medium	

Understand the shape of the dataset

In [11]:
```
data.shape ## Complete the code to get the shape of data
```

Out[11]:
```
(4612, 15)
```

Check the data types of the columns for the dataset

In [13]:
```
# Display the data types of the columns in the dataset
data_types = data.dtypes
data_types
```

Out[13]:
```
ID                        object
age                        int64
current_occupation        object
first_interaction         object
profile_completed         object
website_visits             int64
time_spent_on_website      int64
page_views_per_visit     float64
last_activity             object
print_media_type1         object
print_media_type2         object
digital_media             object
educational_channels      object
referral                  object
status                     int64
dtype: object
```

In [14]:
```python
# Checking for duplicate values in the dataset
duplicate_entries = data.duplicated().sum()
duplicate_entries
```

Out[14]:
```
0
```

Exploratory Data Analysis

Let's check the statistical summary of the data.

In [17]:
```python
# Printing the statistical summary of the data
statistical_summary = data.describe()
statistical_summary
```

Out[17]:

	age	website_visits	time_spent_on_website	page_views_per_visit	
count	4612.00000	4612.00000	4612.00000	4612.00000	4612
mean	46.20121	3.56678	724.01127	3.02613	
std	13.16145	2.82913	743.82868	1.96812	
min	18.00000	0.00000	0.00000	0.00000	
25%	36.00000	2.00000	148.75000	2.07775	
50%	51.00000	3.00000	376.00000	2.79200	
75%	57.00000	5.00000	1336.75000	3.75625	
max	63.00000	30.00000	2537.00000	18.43400	

In [18]:
```python
# Making a list of all categorical variables
cat_col = list(data.select_dtypes("object").columns)

# Printing the count of each unique value in each categorical column
unique_value_counts = {}
```

```
for column in cat_col:
    unique_value_counts[column] = data[column].value_counts()

unique_value_counts
```

```
Out[18]: {'ID': ID
          EXT001     1
          EXT2884    1
          EXT3080    1
          EXT3079    1
          EXT3078    1
                    ..
          EXT1537    1
          EXT1536    1
          EXT1535    1
          EXT1534    1
          EXT4612    1
          Name: count, Length: 4612, dtype: int64,
          'current_occupation': current_occupation
          Professional    2616
          Unemployed      1441
          Student          555
          Name: count, dtype: int64,
          'first_interaction': first_interaction
          Website       2542
          Mobile App    2070
          Name: count, dtype: int64,
          'profile_completed': profile_completed
          High      2264
          Medium    2241
          Low        107
          Name: count, dtype: int64,
          'last_activity': last_activity
          Email Activity      2278
          Phone Activity      1234
          Website Activity    1100
          Name: count, dtype: int64,
          'print_media_type1': print_media_type1
          No     4115
          Yes     497
          Name: count, dtype: int64,
          'print_media_type2': print_media_type2
          No     4379
          Yes     233
          Name: count, dtype: int64,
          'digital_media': digital_media
          No     4085
          Yes     527
          Name: count, dtype: int64,
          'educational_channels': educational_channels
          No     3907
          Yes     705
          Name: count, dtype: int64,
          'referral': referral
          No     4519
          Yes      93
          Name: count, dtype: int64}
```

```
In [19]: # Checking the number of unique values in the "ID" column
         unique_id_count = data["ID"].nunique()
```

```
unique_id_count
```

Out[19]:
```
4612
```

In [20]:
```
# Dropping the "ID" column from the dataset
data.drop(["ID"], axis=1, inplace=True)
```

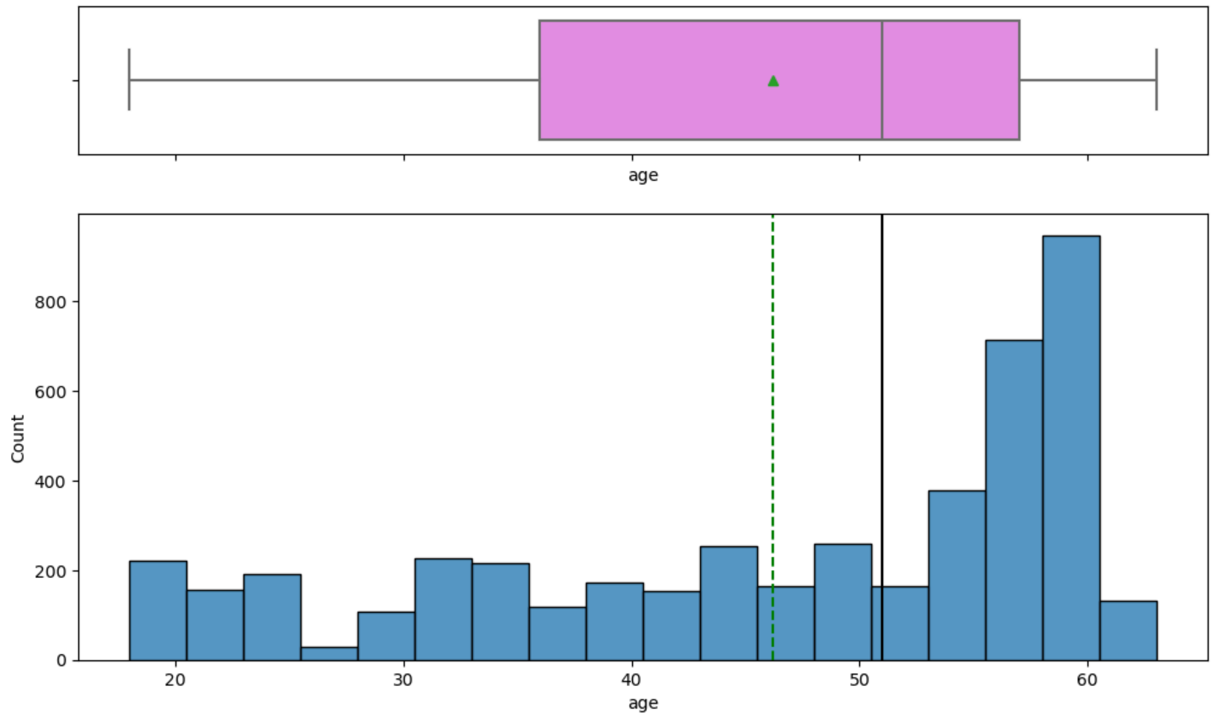
Univariate Analysis

In [22]:
```
# function to plot a boxplot and a histogram along the same scale.


def histogram_boxplot(data, feature, figsize=(12, 7), kde=False, bins=None):
    """
    Boxplot and histogram combined

    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (12,7))
    kde: whether to the show density curve (default False)
    bins: number of bins for histogram (default None)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows=2,  # Number of rows of the subplot grid= 2
        sharex=True,  # x-axis will be shared among all subplots
        gridspec_kw={"height_ratios": (0.25, 0.75)},
        figsize=figsize,
    )  # creating the 2 subplots
    sns.boxplot(
        data=data, x=feature, ax=ax_box2, showmeans=True, color="violet"
    )  # boxplot will be created and a star will indicate the mean value of
    sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2, bins=bins, palette="wint
    ) if bins else sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2
    )  # For histogram
    ax_hist2.axvline(
        data[feature].mean(), color="green", linestyle="--"
    )  # Add mean to the histogram
    ax_hist2.axvline(
        data[feature].median(), color="black", linestyle="-"
    )  # Add median to the histogram
```

Observations on age

- The distribution of age shows a somewhat uniform distribution, with a slight concentration around the age of 50.
- The boxplot indicates a median age of around 51, with the mean also close to this value.
- There are no significant outliers in the age distribution.

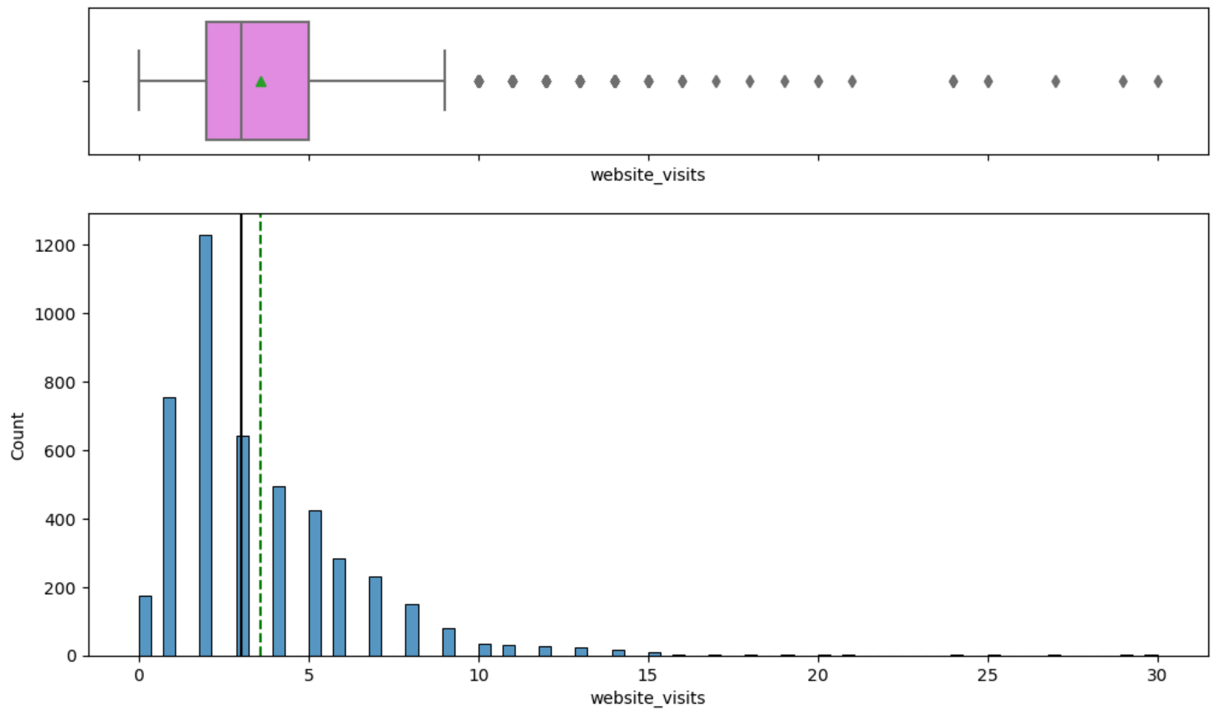
```
In [24]: histogram_boxplot(data, "age")
```



Observations on website_visits

- The distribution of website visits is right-skewed, with most users visiting the website between 2 to 5 times.
- The median number of visits is 3, with the mean slightly higher due to the skewness in the data.
- A few outliers exist where the number of website visits is significantly higher than the majority, reaching up to 30 visits.

```
In [26]: histogram_boxplot(data, "website_visits") # Complete the code to plot a hist
```



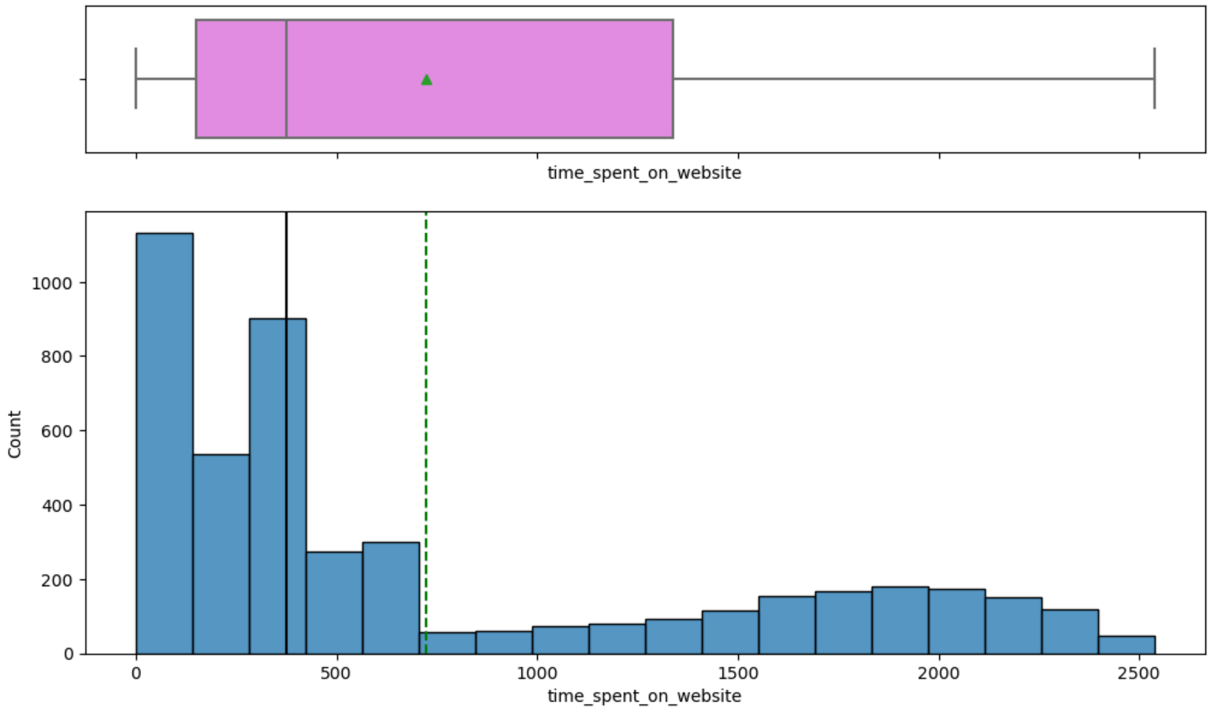
In [27]:
```
# To check how many leads have not visited web-site
data[data["website_visits"] == 0].shape
```

Out[27]:
```
(174, 14)
```

Observations on number of time_spent_on_website

- The distribution of time spent on the website is heavily right-skewed, with the majority of users spending relatively little time on the website.
- The median time spent is much lower than the mean, indicating the presence of a few users who spend a significantly higher amount of time on the website.
- There are several outliers, with a few users spending up to 2537 seconds on the website.

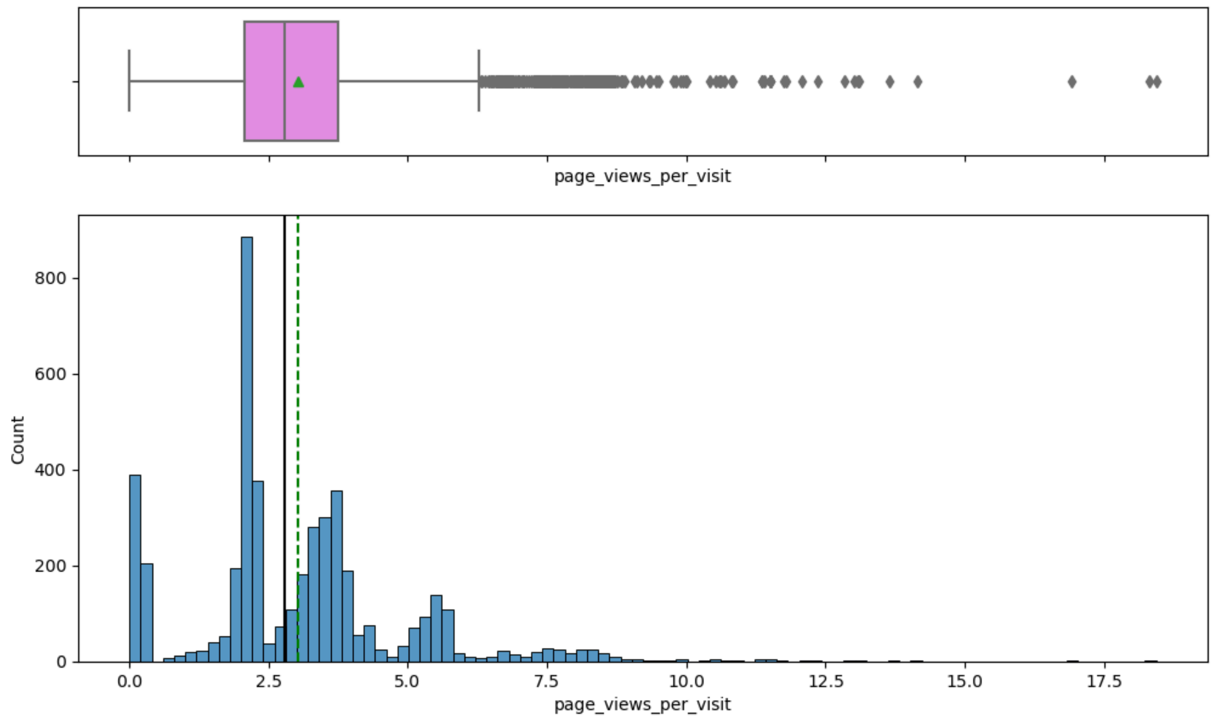
In [29]:
```
histogram_boxplot(data, "time_spent_on_website") # Complete the code to plot
```



Observations on number of page_views_per_visit

- The distribution of page views per visit is also right-skewed, with most users viewing around 2 to 4 pages per visit.
- The median and mean are relatively close, but the presence of outliers causes the mean to be slightly higher.
- A few users view a very high number of pages per visit, which is reflected in the right tail of the distribution.

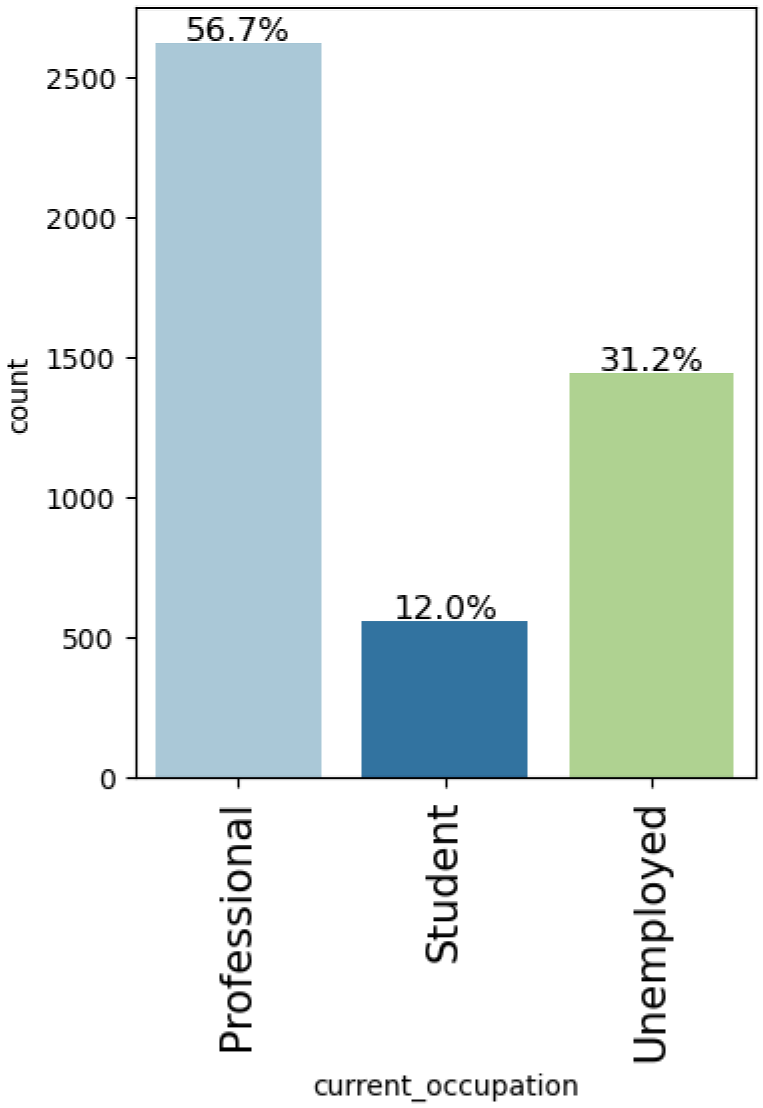
In [31]:
```
histogram_boxplot(data, "page_views_per_visit") # Complete the code to plot
```



In [32]:
```
# function to create labeled barplots


def labeled_barplot(data, feature, perc=False, n=None):
    """
    Barplot with percentage at the top

    data: dataframe
    feature: dataframe column
    perc: whether to display percentages instead of count (default is False)
    n: displays the top n category levels (default is None, i.e., display al
    """

    total = len(data[feature])  # length of the column
    count = data[feature].nunique()
    if n is None:
        plt.figure(figsize=(count + 1, 5))
    else:
        plt.figure(figsize=(n + 1, 5))

    plt.xticks(rotation=90, fontsize=15)
    ax = sns.countplot(
        data=data,
        x=feature,
        palette="Paired",
        order=data[feature].value_counts().index[:n].sort_values(),
    )

    for p in ax.patches:
        if perc == True:
            label = "{:.1f}%".format(
                100 * p.get_height() / total
            )  # percentage of each class of the category
        else:
            label = p.get_height()  # count of each level of the category

        x = p.get_x() + p.get_width() / 2  # width of the plot
        y = p.get_height()  # height of the plot

        ax.annotate(
            label,
            (x, y),
            ha="center",
            va="center",
            size=12,
            xytext=(0, 5),
            textcoords="offset points",
        )  # annotate the percentage

    plt.show()  # show the plot
```

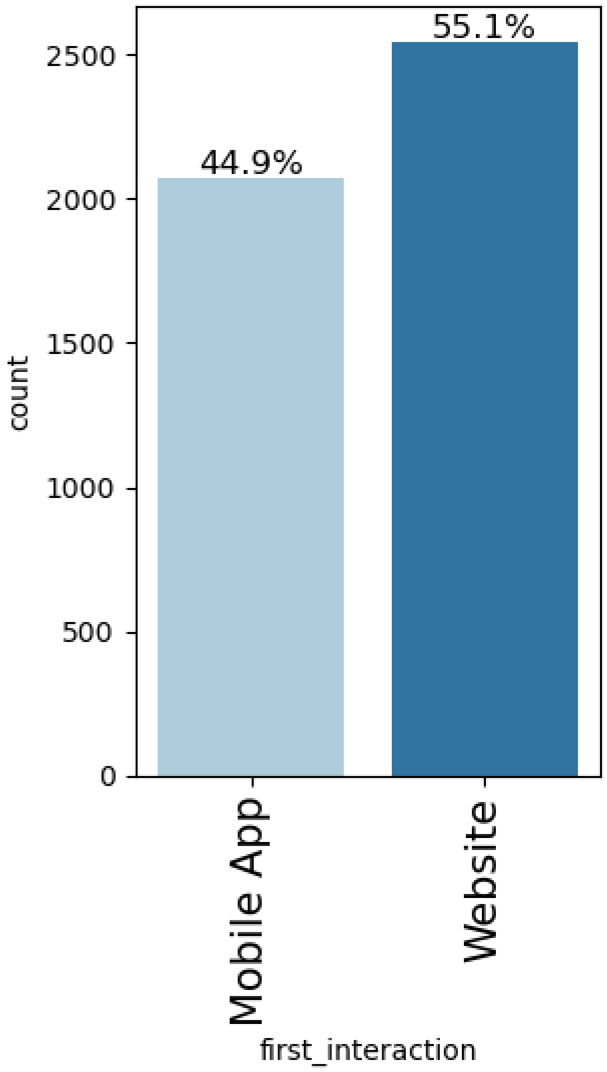
Observations on current_occupation

In [34]:
```
labeled_barplot(data, "current_occupation", perc=True)
```



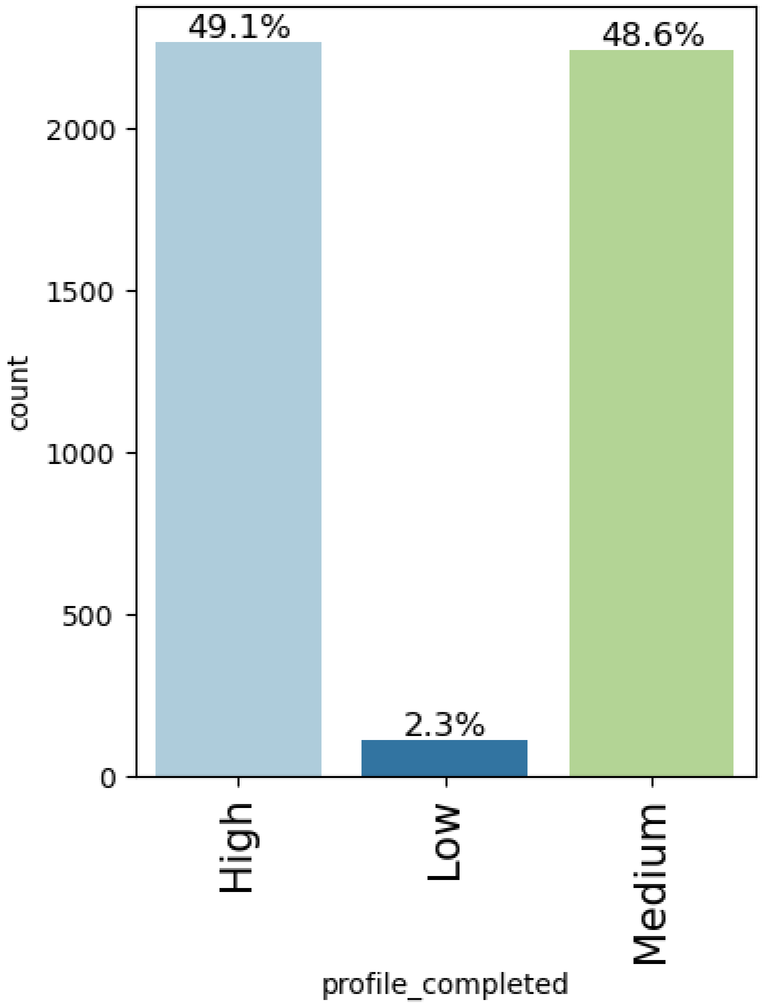
Observations on number of first_interaction

```
In [36]: labeled_barplot(data, "first_interaction", perc=True) # Complete the code to
```



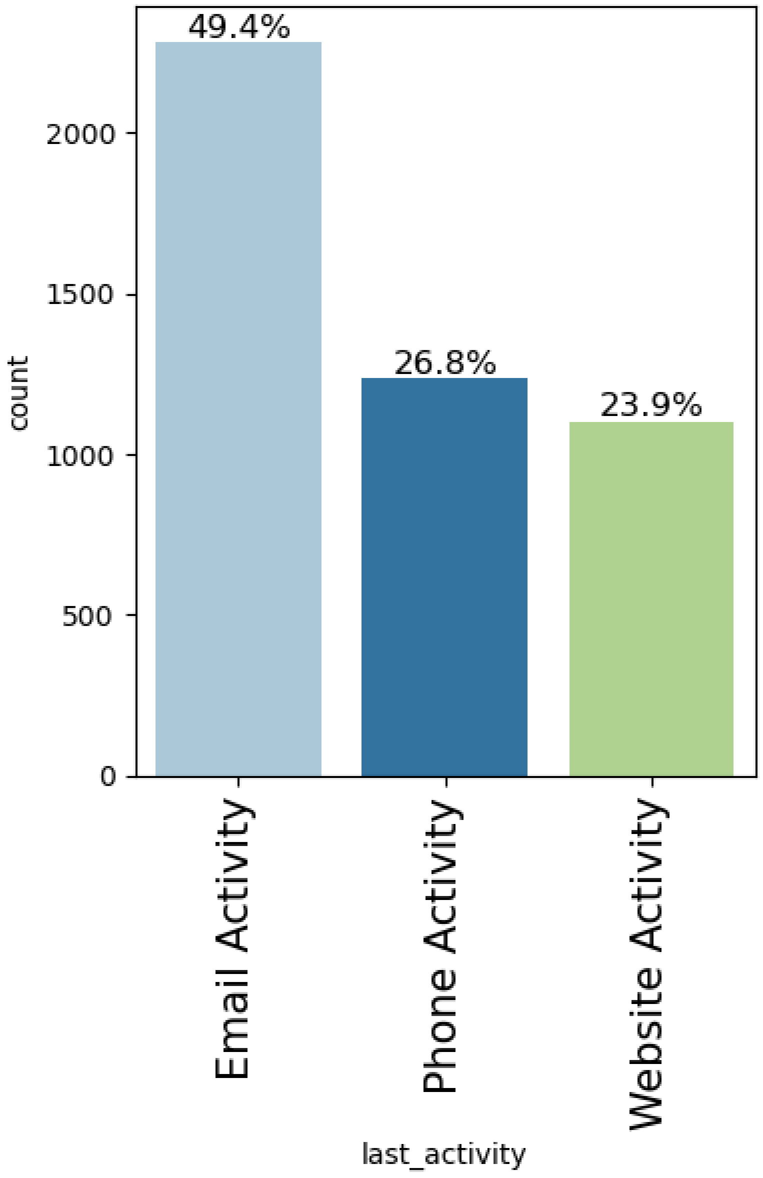
Observations on profile_completed

```
In [38]: labeled_barplot(data, "profile_completed", perc=True) # Complete the code to
```



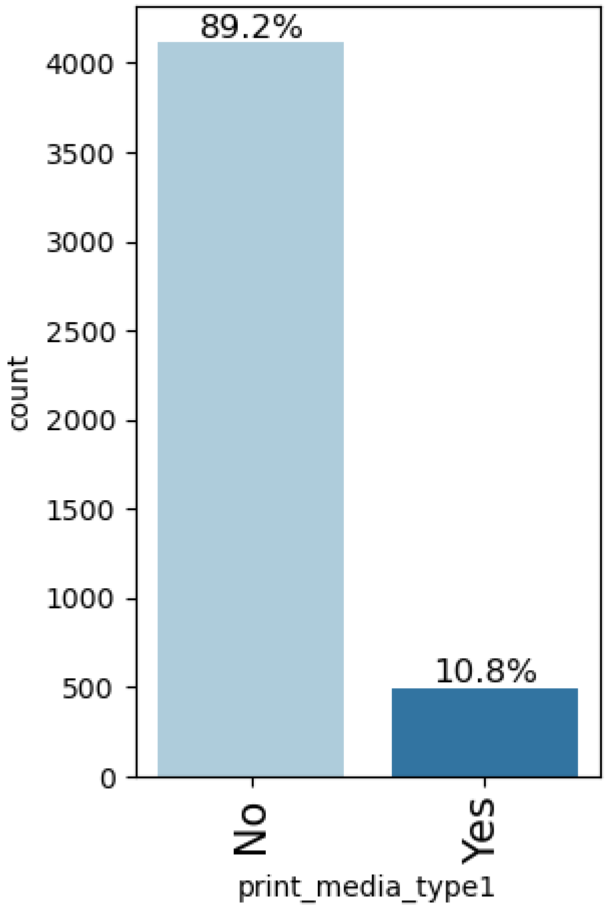
Observations on last_activity

In [40]:
```
labeled_barplot(data, "last_activity", perc=True) # Complete the code to plo
```



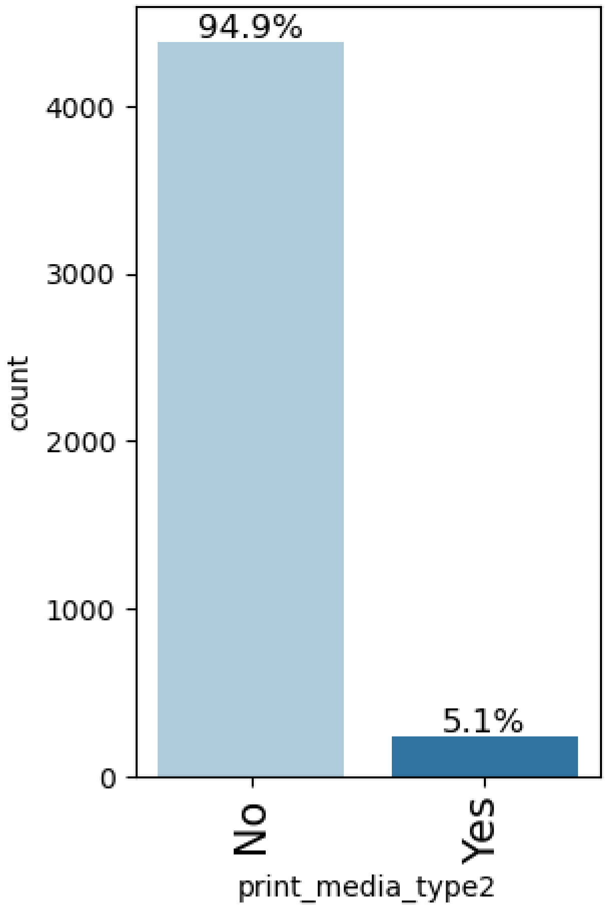
Observations on print_media_type1

```
In [42]: labeled_barplot(data, "print_media_type1", perc=True) # Complete the code tc
```



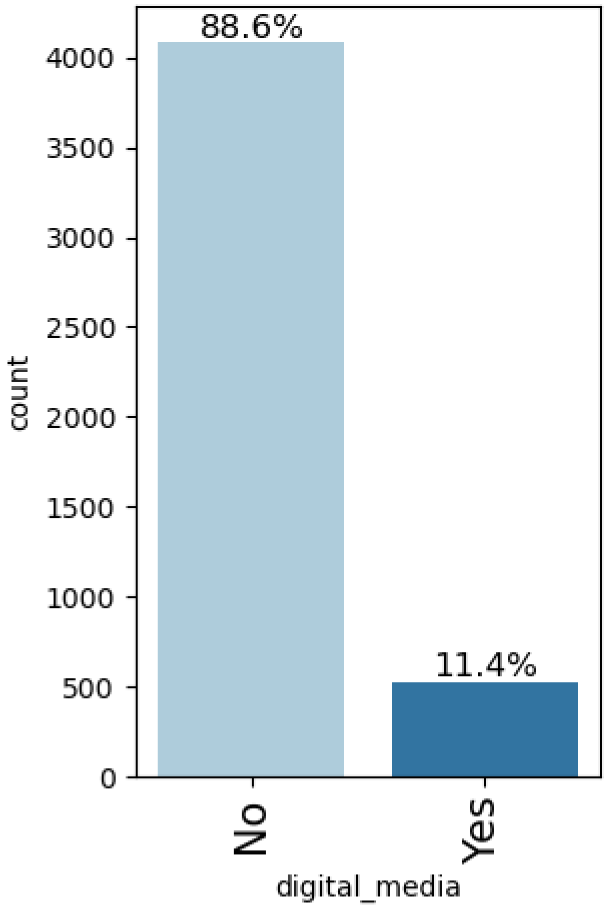
Observations on print_media_type2

In [44]:
```
labeled_barplot(data, "print_media_type2", perc=True) # Complete the code to
```



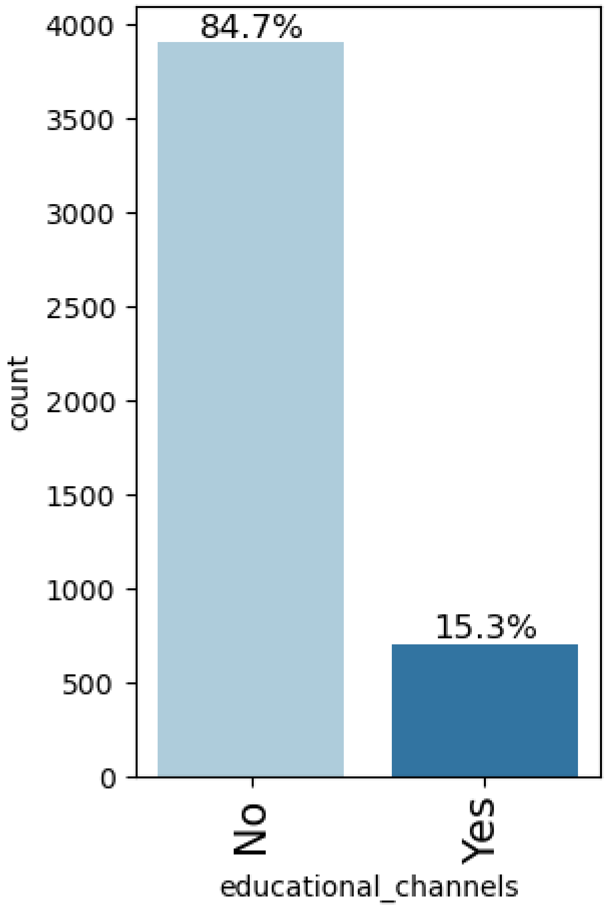
Observations on digital_media

In [46]:
```
labeled_barplot(data, "digital_media", perc=True) # Complete the code to plo
```



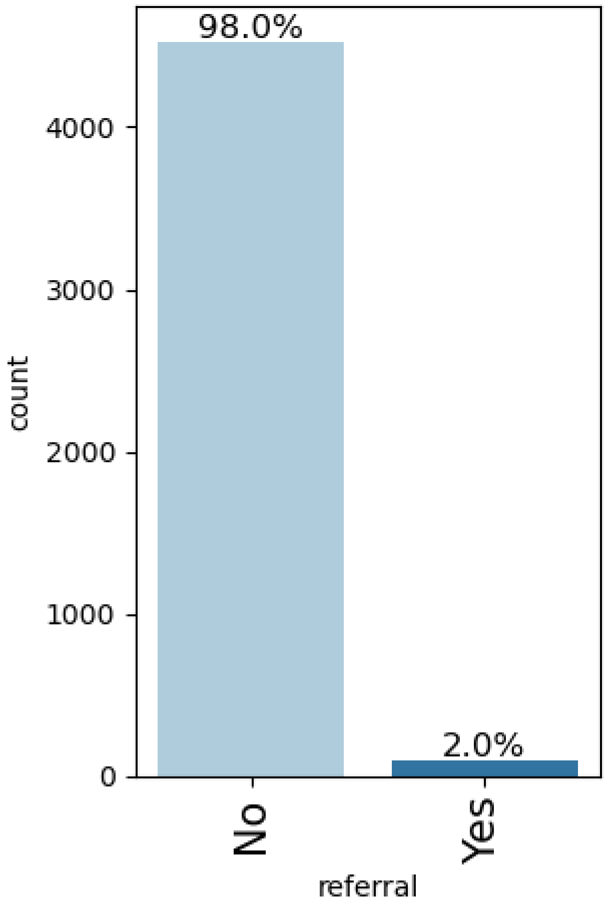
Observations on educational_channels

In [48]:
```
labeled_barplot(data, "educational_channels", perc=True) # Complete the code
```



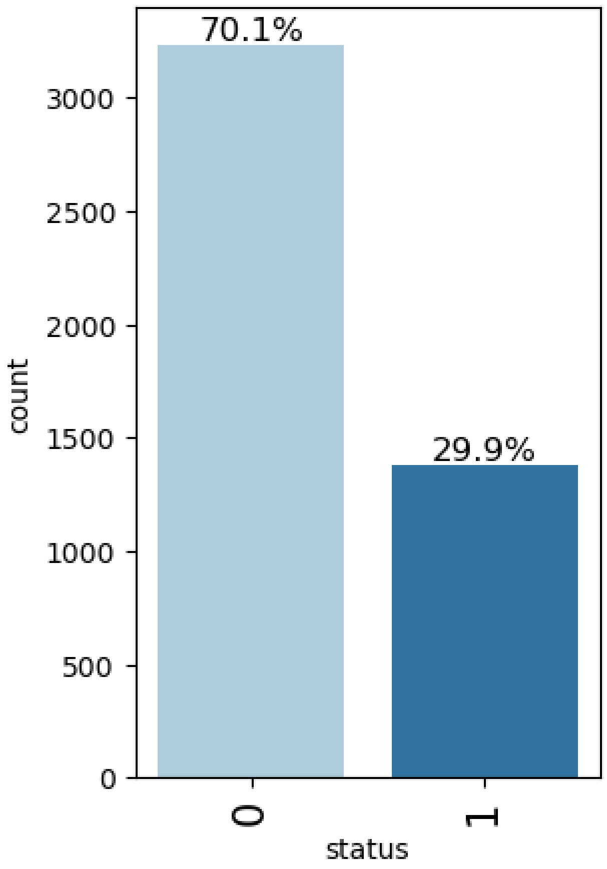
Observations on referral

```
In [50]: labeled_barplot(data, "referral", perc=True) # Complete the code to plot lab
```



Observations on status

```
In [52]: labeled_barplot(data, "status", perc=True) # Complete the code to plot label
```



Observations from Univariate Analysis:

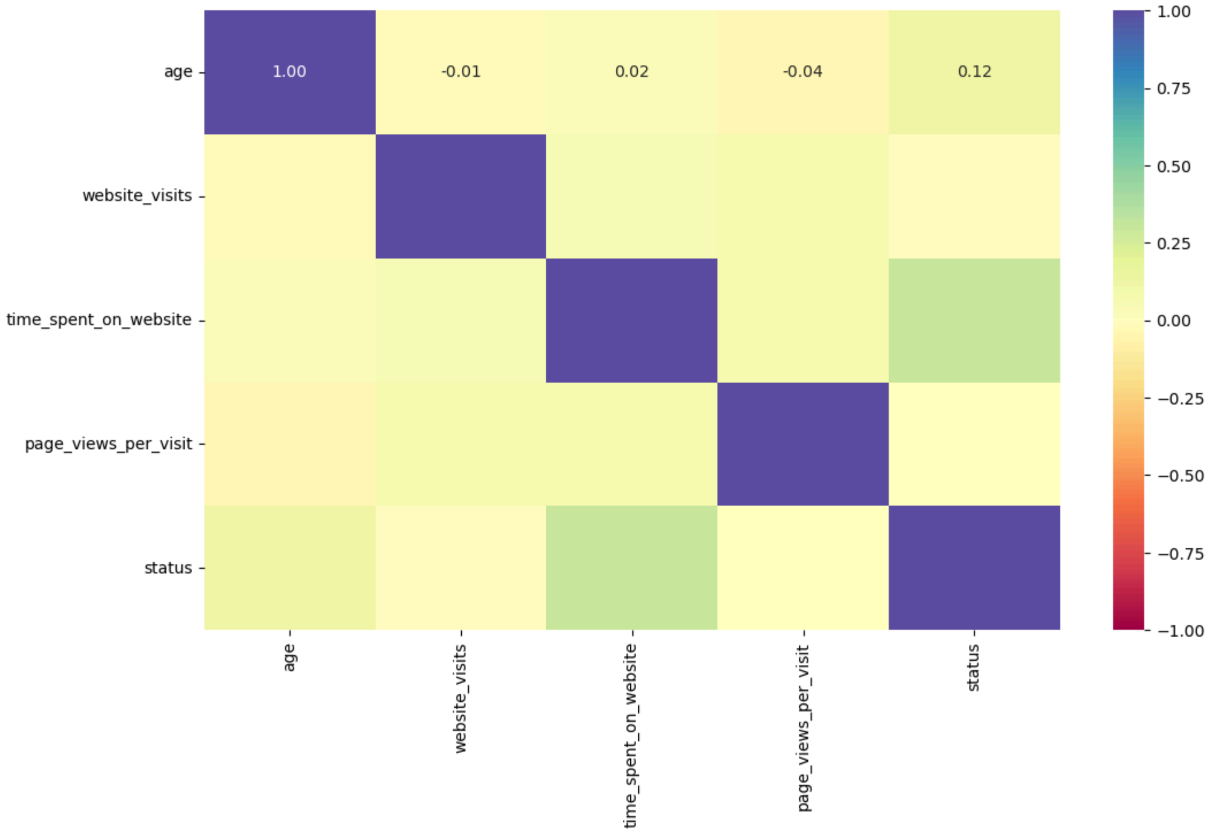
```
1.      First Interaction:
•       The majority of users first interacted with the
platform via the website (about 55%), with the remaining
using the mobile app (45%).
2.      Profile Completed:
•       Most users have a “High” or “Medium” level of profile
completion, with “High” slightly more common. Very few users
have a “Low” profile completion level.
3.      Last Activity:
•       The last activity of most users is related to “Email
Activity” or “Phone Activity,” with “Website Activity” being
the least common.
4.      Print Media Type 1:
•       The majority of users did not engage with Print Media
Type 1, with a small portion (around 10%) who did.
5.      Print Media Type 2:
•       Even fewer users engaged with Print Media Type 2,
with only about 5% of users interacting with it.
6.      Digital Media:
•       Around 11% of users engaged with Digital Media, while
the vast majority (89%) did not.
7.      Educational Channels:
•       Most users (85%) did not engage with Educational
Channels, while a minority did (15%).
```

```
8.      Referral:
•       A very small percentage of users (around 2%) were
referred by others, while the overwhelming majority were not.
9.      Status:
•       The “status” variable, which indicates whether a user
is a potential customer, shows that around 30% of the users
are potential customers, while 70% are not.
```

These distributions provide a clear picture of user behavior and engagement with different media and channels, as well as their interaction and completion levels on the platform. Understanding these patterns is crucial for tailoring strategies to engage more users and convert them into potential customers.

Bivariate Analysis

In [55]:
```python
cols_list = data.select_dtypes(include=np.number).columns.tolist()

plt.figure(figsize=(12, 7))
sns.heatmap(
    data[cols_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Sp
)
plt.show()
```



Creating functions that will help us with further analysis.

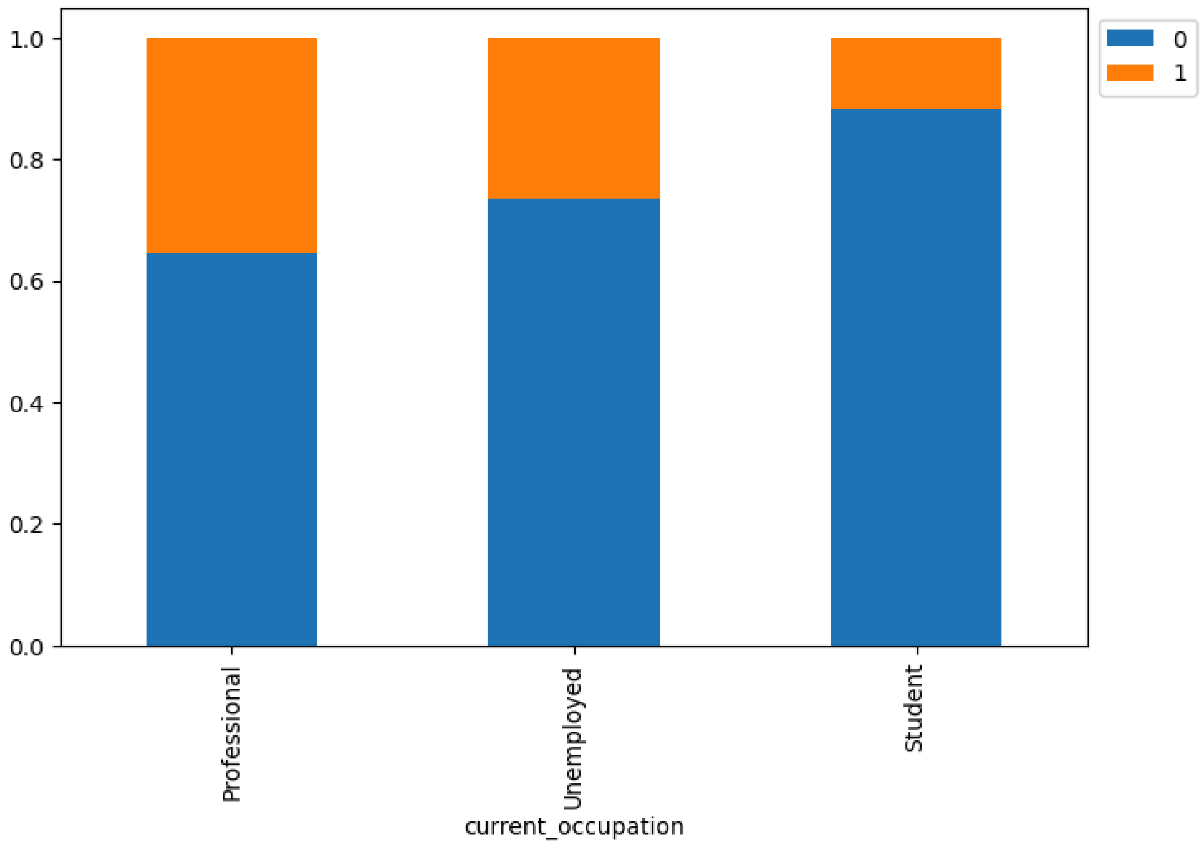
In [57]:
```
### function to plot distributions wrt target


def distribution_plot_wrt_target(data, predictor, target):

    fig, axs = plt.subplots(2, 2, figsize=(12, 10))

    target_uniq = data[target].unique()

    axs[0, 0].set_title("Distribution of target for target=" + str(target_ur
    sns.histplot(
        data=data[data[target] == target_uniq[0]],
        x=predictor,
        kde=True,
        ax=axs[0, 0],
        color="teal",
        stat="density",
    )

    axs[0, 1].set_title("Distribution of target for target=" + str(target_ur
    sns.histplot(
        data=data[data[target] == target_uniq[1]],
        x=predictor,
        kde=True,
        ax=axs[0, 1],
        color="orange",
        stat="density",
    )

    axs[1, 0].set_title("Boxplot w.r.t target")
    sns.boxplot(data=data, x=target, y=predictor, ax=axs[1, 0], palette="gis

    axs[1, 1].set_title("Boxplot (without outliers) w.r.t target")
    sns.boxplot(
        data=data,
        x=target,
        y=predictor,
        ax=axs[1, 1],
        showfliers=False,
        palette="gist_rainbow",
    )

    plt.tight_layout()
    plt.show()
```

In [58]:
```
def stacked_barplot(data, predictor, target):
    """
    Print the category counts and plot a stacked bar chart

    data: dataframe
    predictor: independent variable
    target: target variable
    """
    count = data[predictor].nunique()
    sorter = data[target].value_counts().index[-1]
    tab1 = pd.crosstab(data[predictor], data[target], margins=True).sort_val
        by=sorter, ascending=False
    )
    print(tab1)
    print("-" * 120)
    tab = pd.crosstab(data[predictor], data[target], normalize="index").sort
        by=sorter, ascending=False
    )
    tab.plot(kind="bar", stacked=True, figsize=(count + 5, 5))
    plt.legend(
        loc="lower left", frameon=False,
    )
    plt.legend(loc="upper left", bbox_to_anchor=(1, 1))
    plt.show()
```

Leads will have different expectations from the outcome of the course and the current occupation may play a key role for them to take the program. Let's analyze it

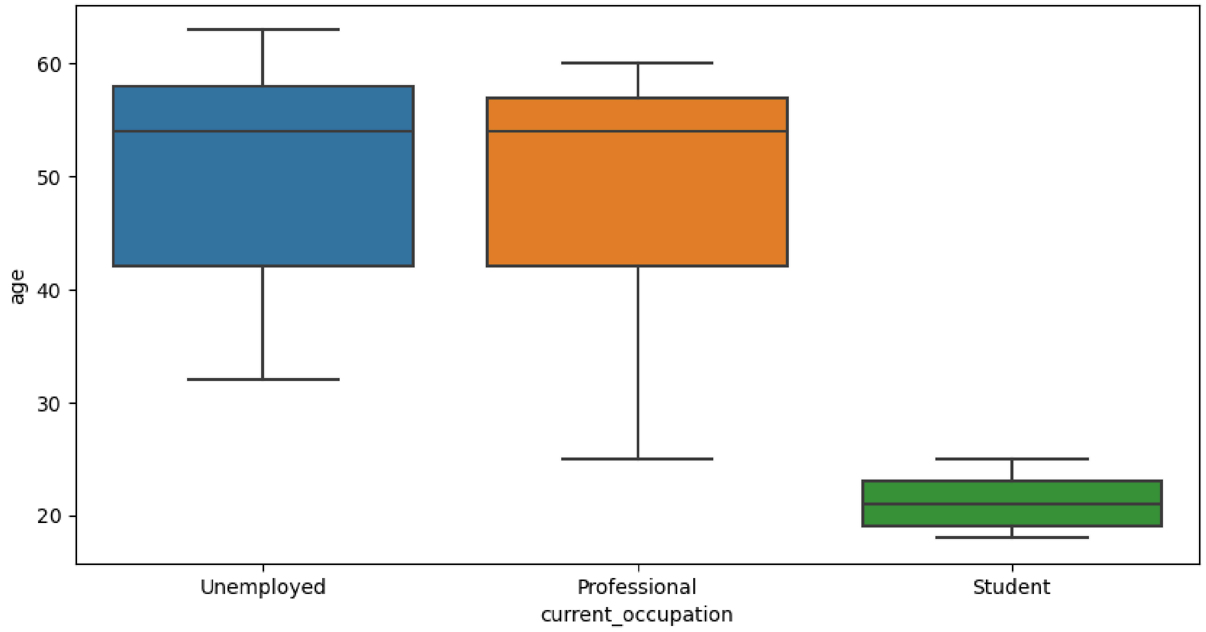
In [60]:
```
stacked_barplot(data, "current_occupation", "status")
```

```
status                 0     1   All
current_occupation
All                 3235  1377  4612
Professional        1687   929  2616
Unemployed          1058   383  1441
Student              490    65   555
------------------------------------------------------------------------------------------------------------------------
```



Age can be a good factor to differentiate between such leads

In [62]:
```
plt.figure(figsize=(10, 5))
sns.boxplot(data = data, x = data["current_occupation"], y = data["age"])
plt.show()
```



In [63]:
```
data.groupby(["current_occupation"])["age"].describe()
```

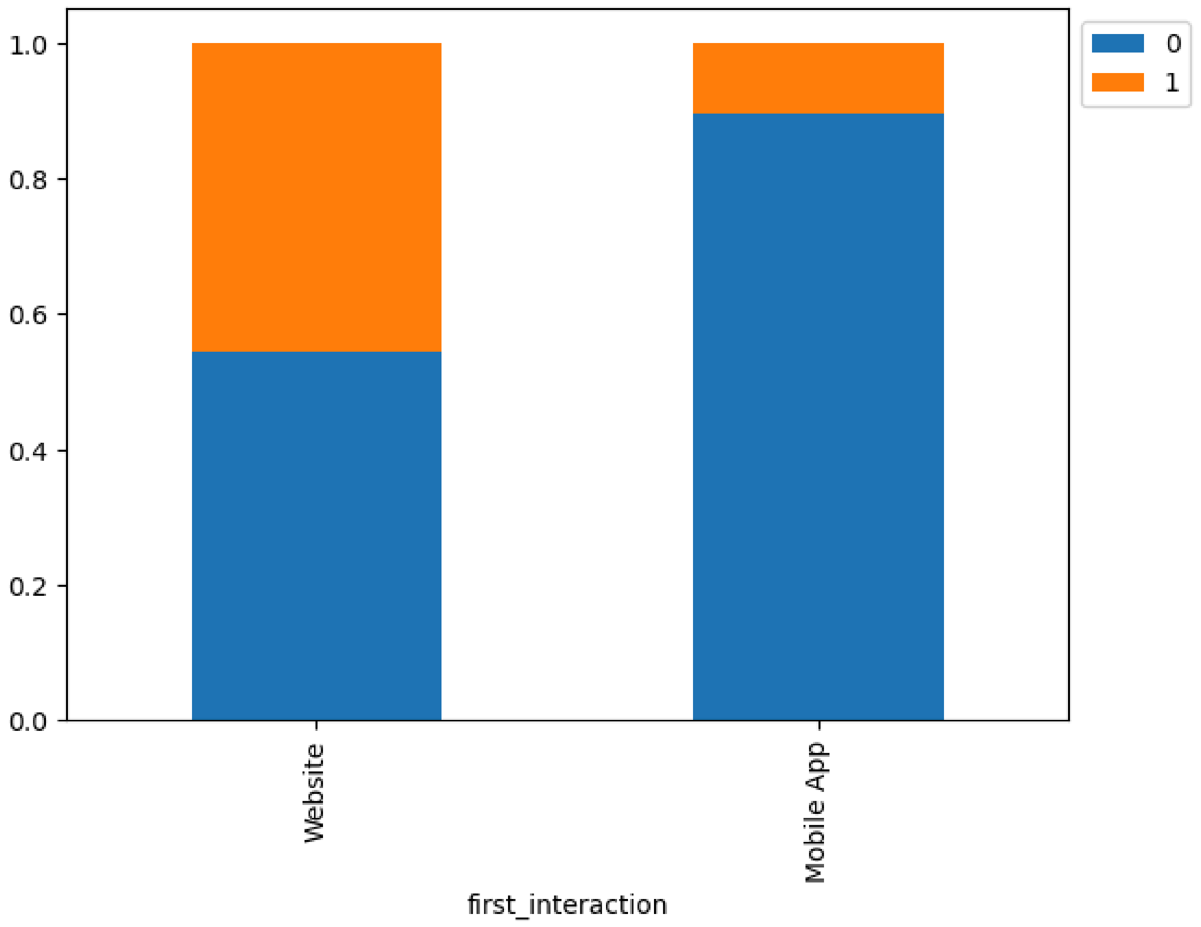
Out[63]:

current_occupation	count	mean	std	min	25%	50%	
Professional	2616.00000	49.34748	9.89074	25.00000	42.00000	54.00000	5
Student	555.00000	21.14414	2.00111	18.00000	19.00000	21.00000	2
Unemployed	1441.00000	50.14018	9.99950	32.00000	42.00000	54.00000	5

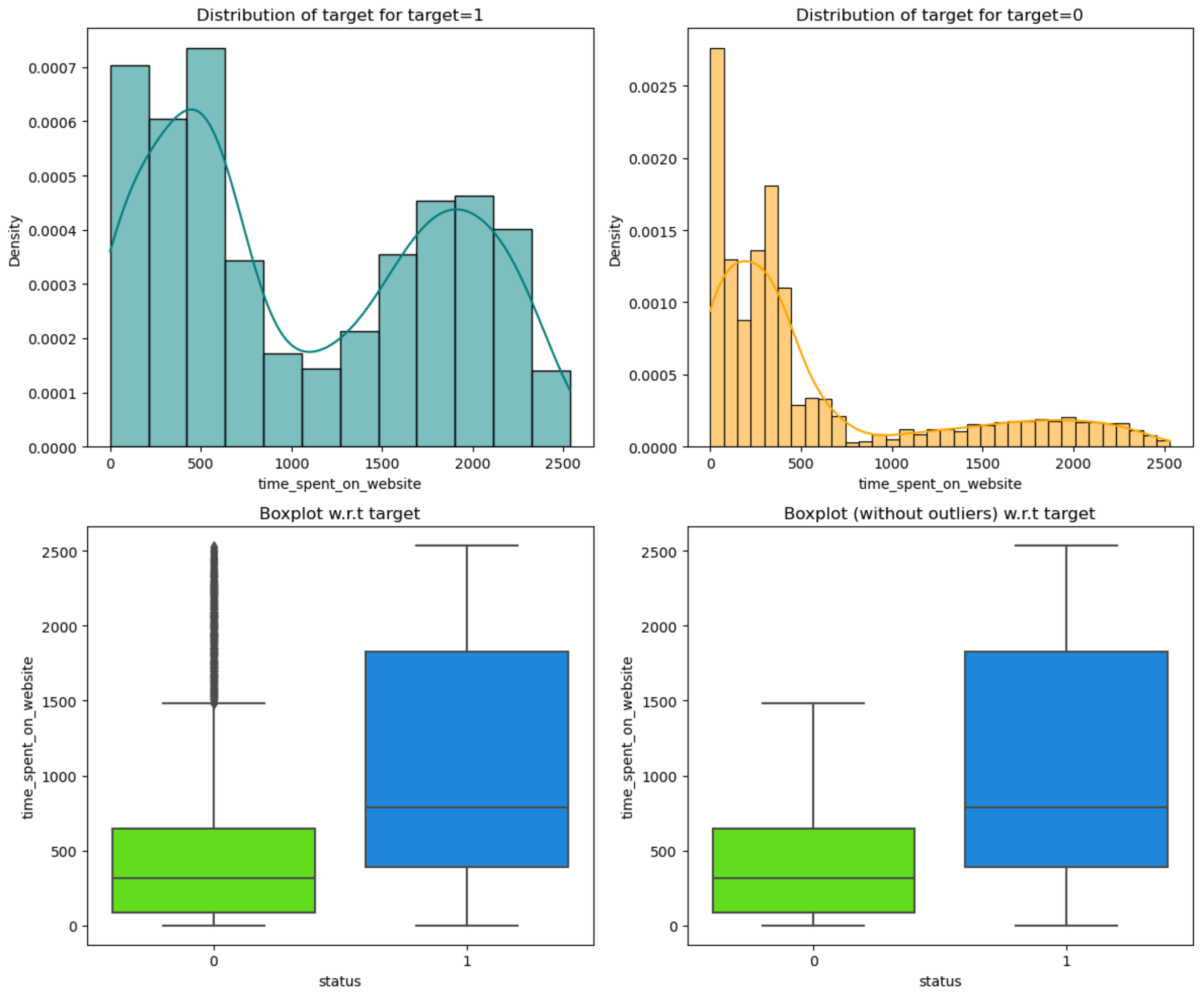
The company's first interaction with leads should be compelling and persuasive. Let's see if the channels of the first interaction have an impact on the conversion of leads

In [65]:
```
stacked_barplot(data, "first_interaction", "status") # Complete the code to
```

```
status                0     1   All
first_interaction
All                3235  1377  4612
Website            1383  1159  2542
Mobile App         1852   218  2070
------------------------------------------------------------------------------
--------------------------------------------
```



In [66]:
```
distribution_plot_wrt_target(data, "time_spent_on_website", "status")
```

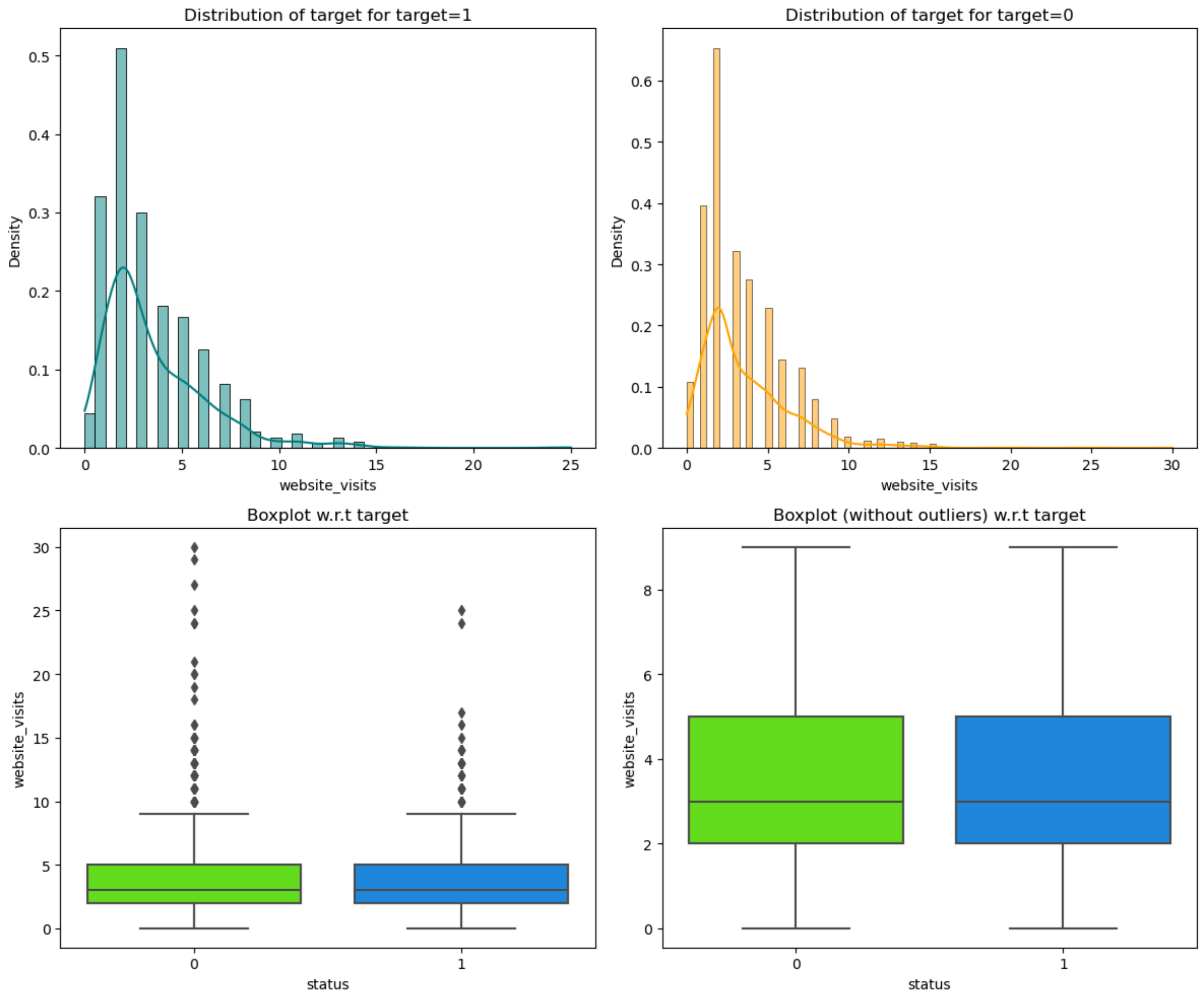


In [67]:
```
# checking the median value
data.groupby(["status"])["time_spent_on_website"].median()
```

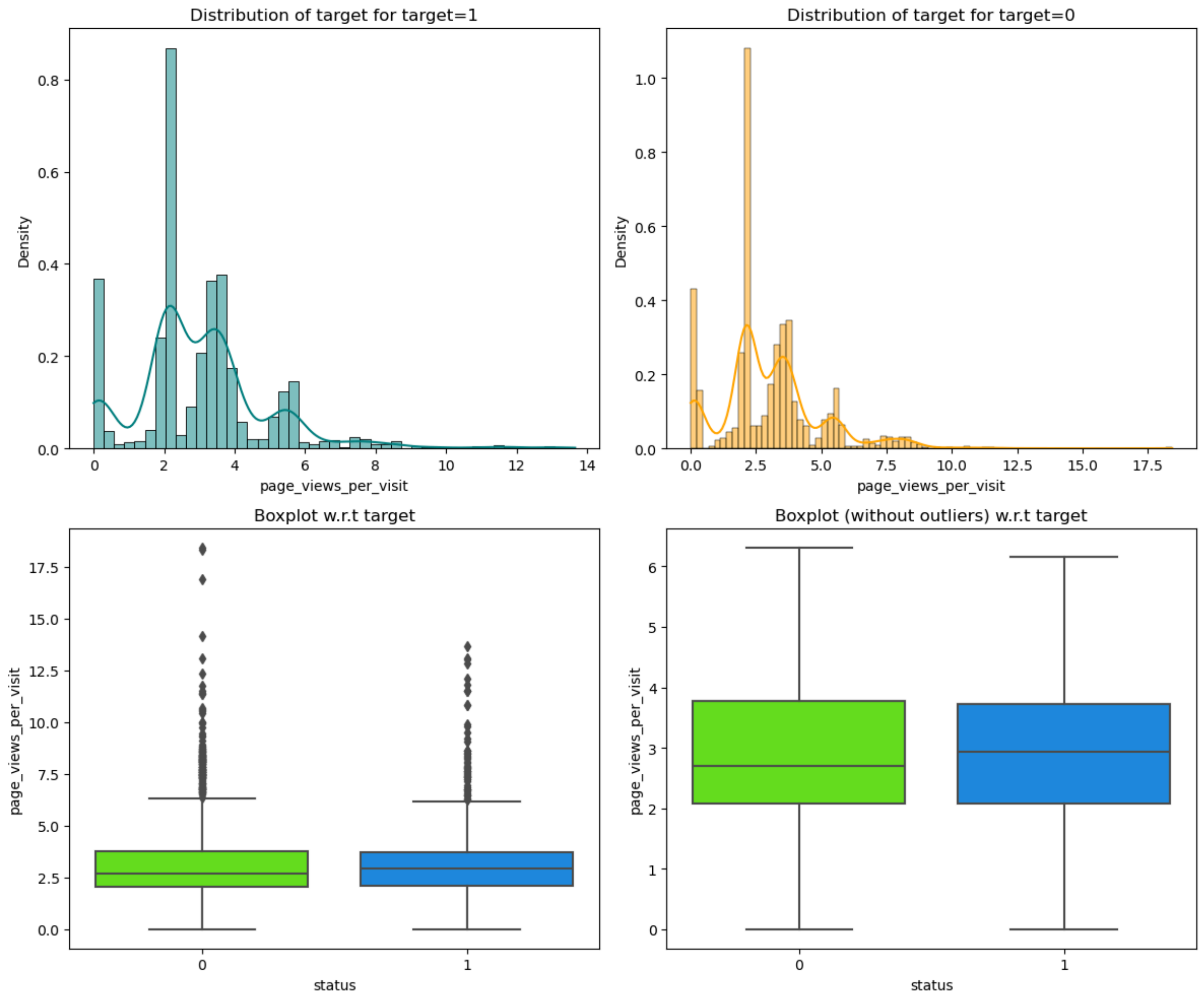
Out[67]:
```
status
0    317.00000
1    789.00000
Name: time_spent_on_website, dtype: float64
```

Let's do a similar analysis for time spent on website and page views per visit.

In [69]:
```
distribution_plot_wrt_target(data, "website_visits", "status") # Complete th
```



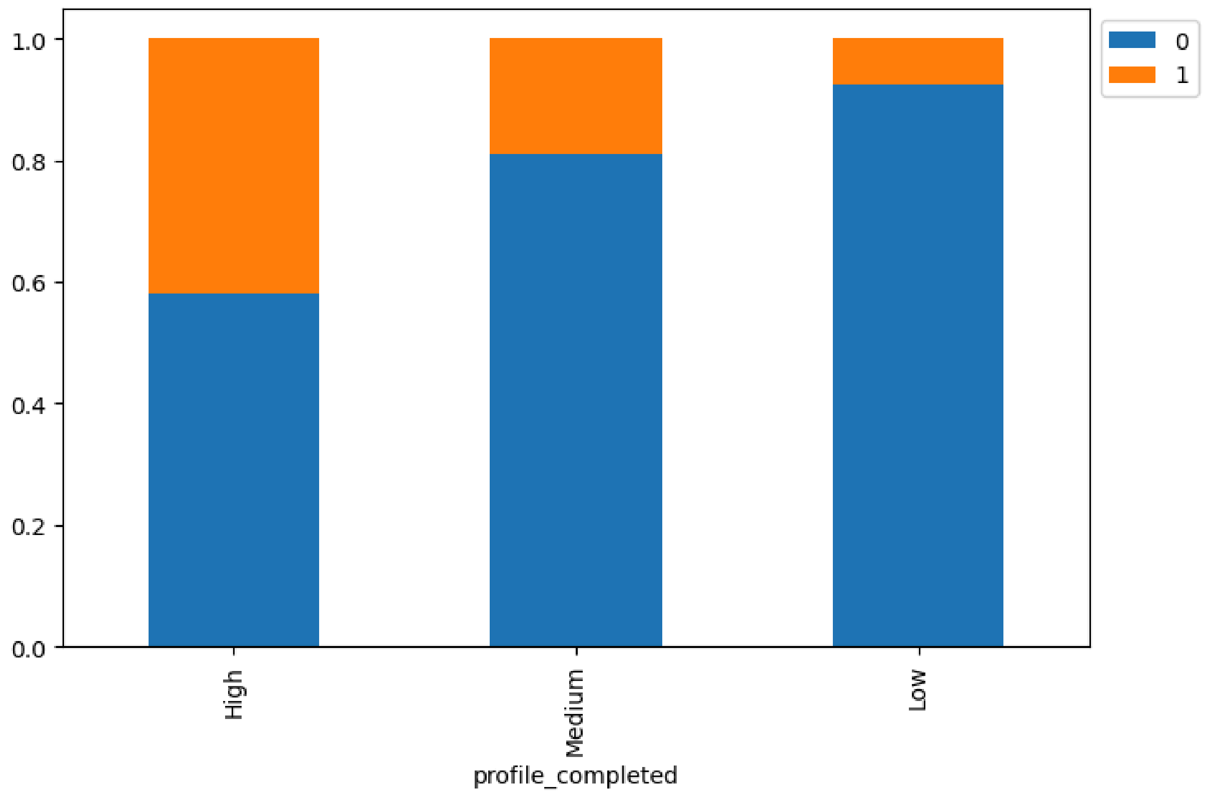
```
In [70]: distribution_plot_wrt_target(data, "page_views_per_visit", "status") # Compl
```



People browsing the website or the mobile app are generally required to create a profile by sharing their personal details before they can access more information. Let's see if the profile completion level has an impact on lead status

In [72]:
```
stacked_barplot(data, "profile_completed", "status")  # Complete the code to
```

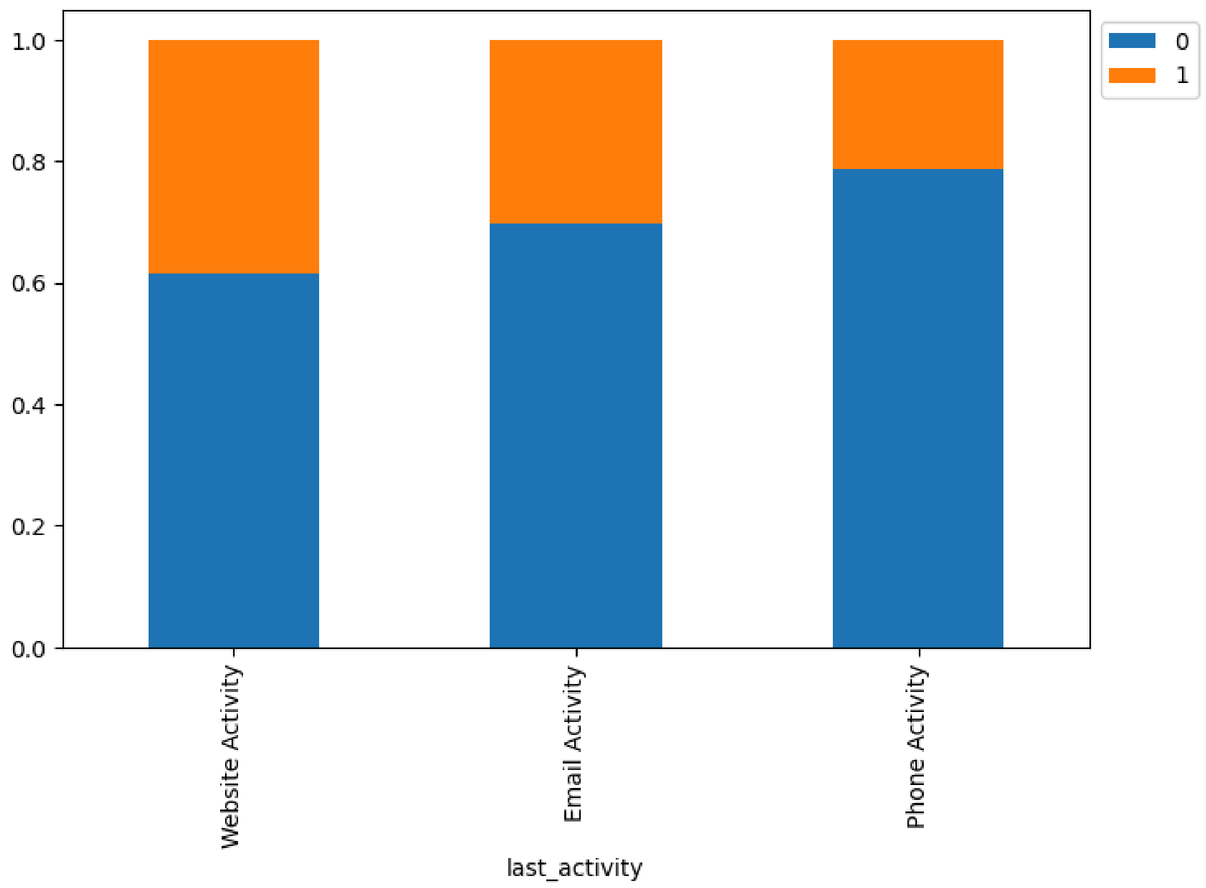
```
status                 0     1   All
profile_completed
All                 3235  1377  4612
High                1318   946  2264
Medium              1818   423  2241
Low                   99     8   107
------------------------------------------------------------------------------
---------------------------------------------
```



After a lead shares their information by creating a profile, there may be interactions between the lead and the company to proceed with the process of enrollment. Let's see how the last activity impacts lead conversion status

In [74]:
```
stacked_barplot(data, "last_activity", "status") # Complete the code to plot
```

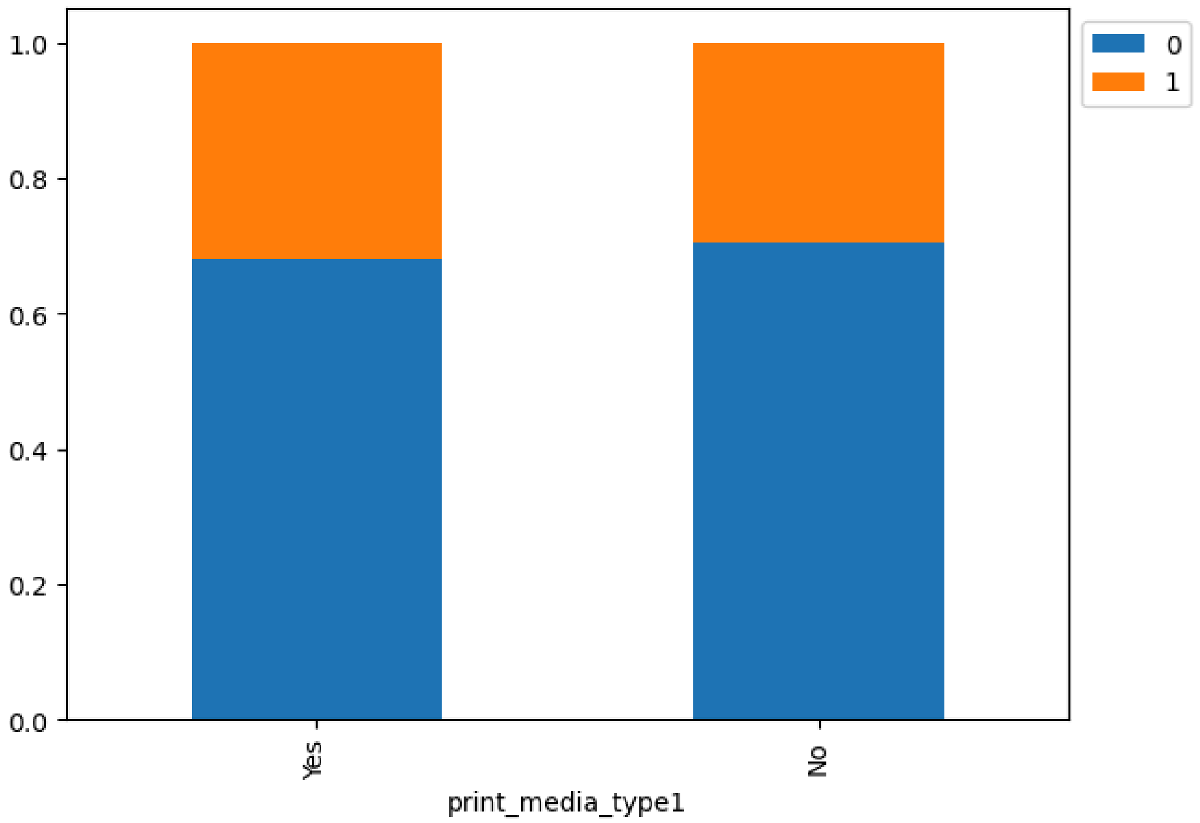
```
status                0     1   All
last_activity
All                3235  1377  4612
Email Activity     1587   691  2278
Website Activity    677   423  1100
Phone Activity      971   263  1234
------------------------------------------------------------------------------
--------------------------------------------
```



Let's see how advertisement and referrals impact the lead status

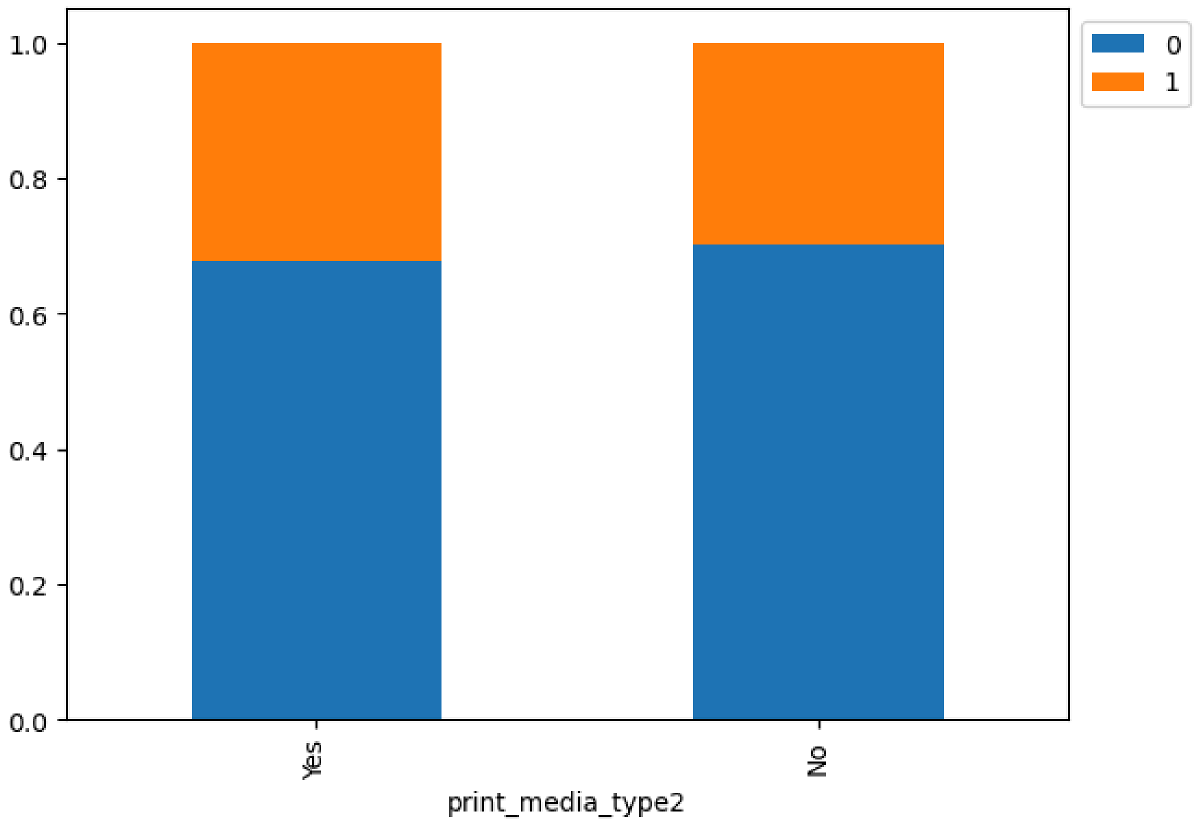
```
In [76]: stacked_barplot(data, "print_media_type1", "status") # Complete the code to
```

```
status               0     1   All
print_media_type1
All               3235  1377  4612
No                2897  1218  4115
Yes                338   159   497
------------------------------------------------------------------------------
---------------------------------------------
```



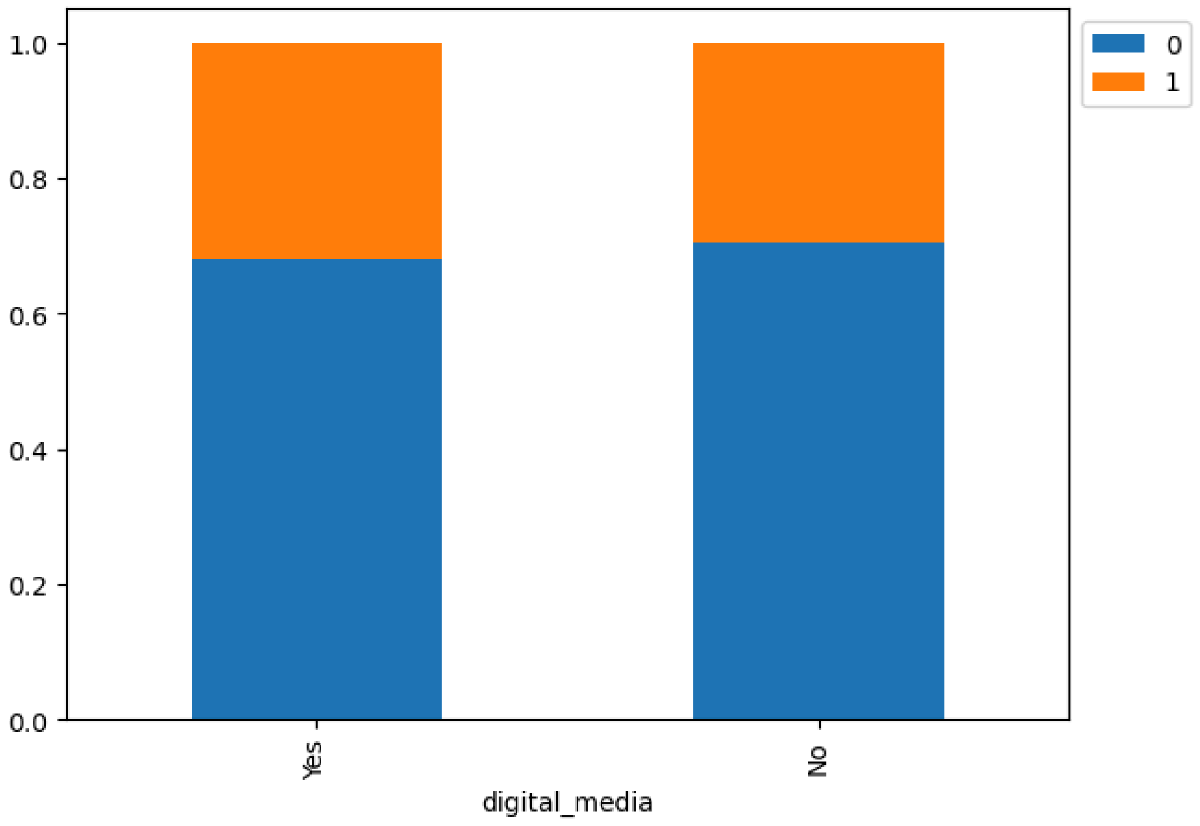
```
In [77]: stacked_barplot(data, "print_media_type2", "status") # Complete the code to
```

```
status                0     1   All
print_media_type2
All                3235  1377  4612
No                 3077  1302  4379
Yes                 158    75   233
------------------------------------------------------------------------------
------------------------------------------
```



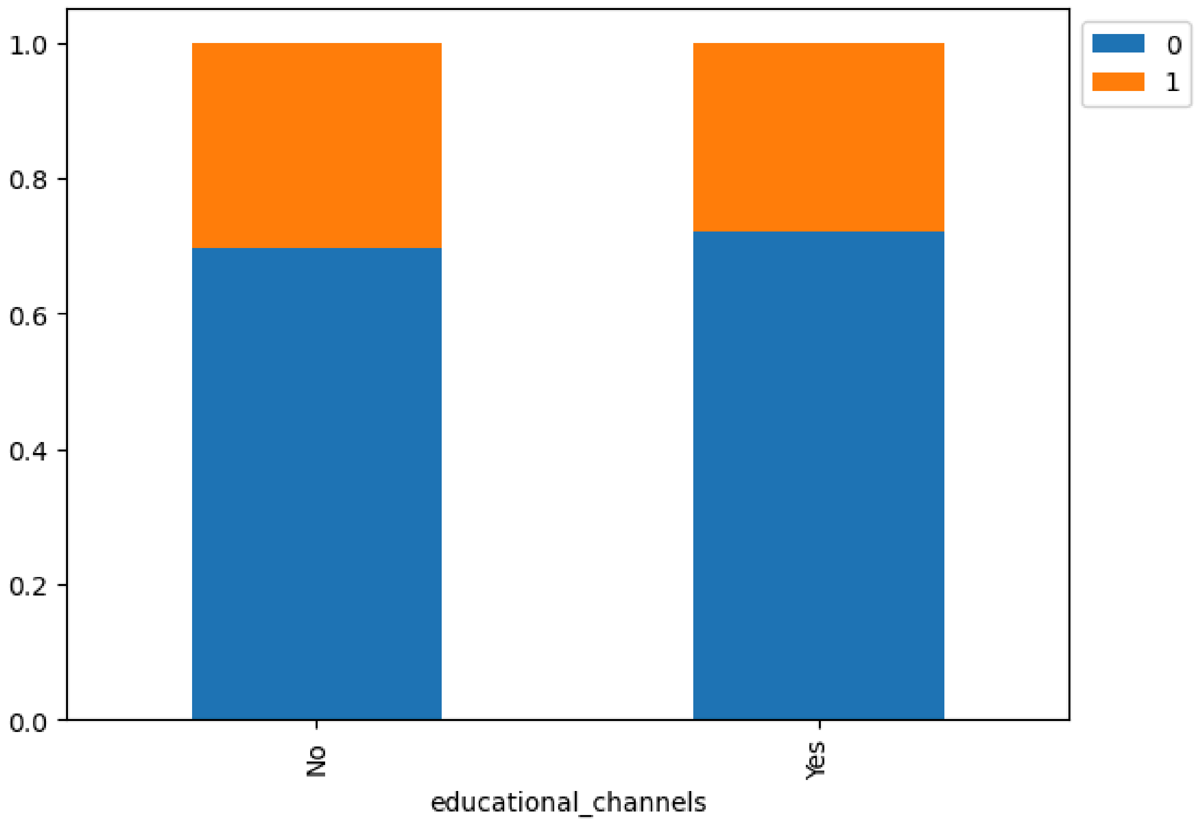
In [78]:
```
stacked_barplot(data, "digital_media", "status") # Complete the code to plot
```

```
status            0     1   All
digital_media
All            3235  1377  4612
No             2876  1209  4085
Yes             359   168   527
------------------------------------------------------------------------------
--------------------------------------------
```



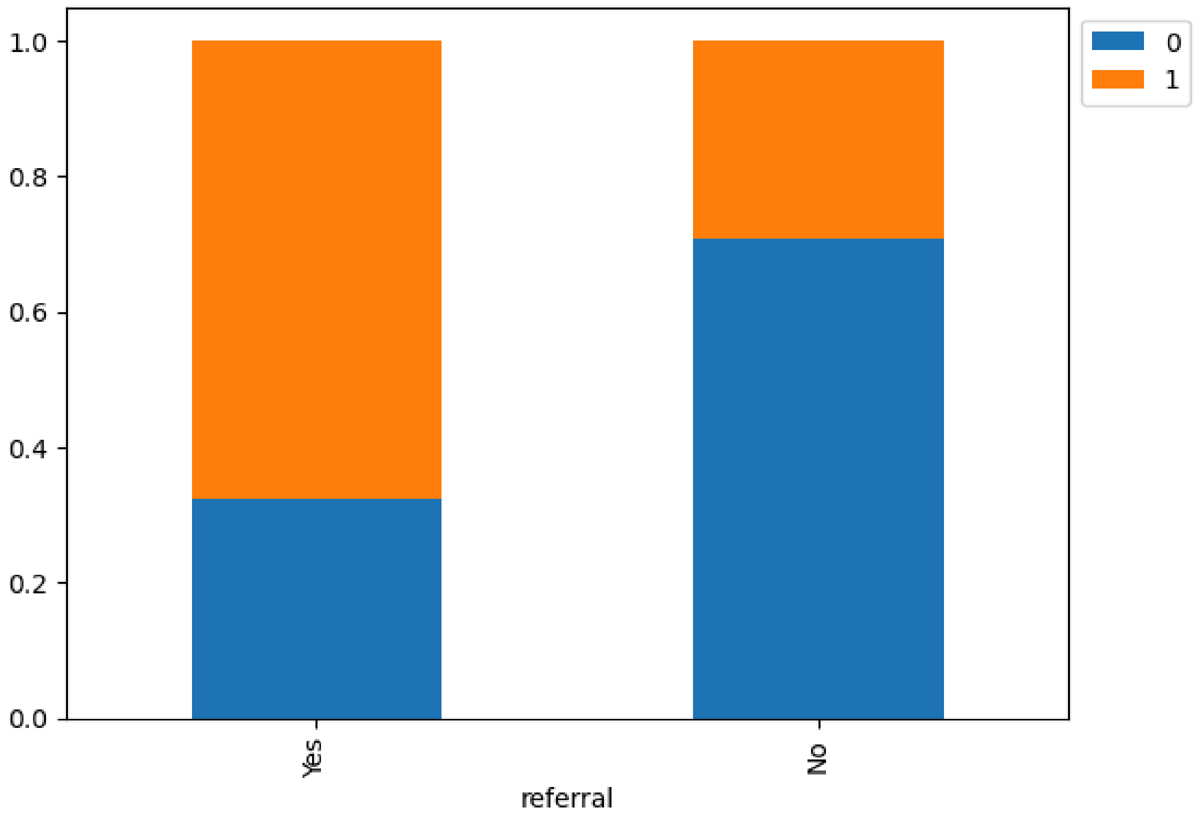
In [79]:
```
stacked_barplot(data, "educational_channels", "status") # Complete the code
```

```
status                   0     1   All
educational_channels                  
All                   3235  1377  4612
No                    2727  1180  3907
Yes                    508   197   705
------------------------------------------------------------------------------------------------------------------------
```



```
In [80]: stacked_barplot(data, "referral", "status") # Complete the code to plot stac
```

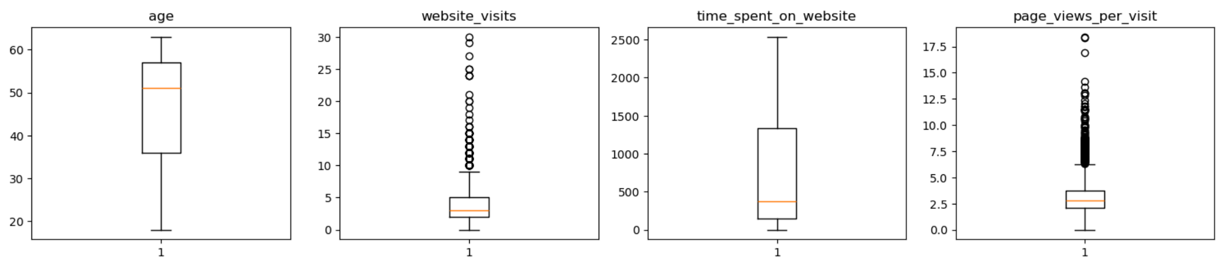
```
status       0     1   All
referral
All       3235  1377  4612
No        3205  1314  4519
Yes         30    63    93
------------------------------------------------------------------------------------------------------------------------
```



Outlier Check

- Let's check for outliers in the data.

In [82]:
```
# outlier detection using boxplot
numeric_columns = data.select_dtypes(include=np.number).columns.tolist()
# dropping release_year as it is a temporal variable
numeric_columns.remove("status")

plt.figure(figsize=(15, 12))

for i, variable in enumerate(numeric_columns):
    plt.subplot(4, 4, i + 1)
    plt.boxplot(data[variable], whis=1.5)
    plt.tight_layout()
    plt.title(variable)

plt.show()
```



Observations:

1. Age: • The boxplot for age shows very few, if any, outliers. The distribution is relatively consistent, with no extreme values.

2. Website Visits: • There are several outliers in the number of website visits. While most users visit the website between 2 and 5 times, some visit up to 30 times, which are considered outliers.

3. Time Spent on Website: • The boxplot indicates significant outliers in the time spent on the website. Most users spend a moderate amount of time, but a few spend a very high amount of time, which deviates significantly from the rest.

4. Page Views per Visit: • There are notable outliers in the number of page views per visit. While most users view around 2 to 4 pages per visit, some users view up to 18 pages, which is an outlier in this context.

Data Preparation for modeling

- We want to predict which lead is more likely to be converted.
- Before we proceed to build a model, we'll have to encode categorical features.
- We'll split the data into train and test to be able to evaluate the model that we build on the train data.

In [85]:
```python
# check the categorical columns and apply one-hot encoding
X = pd.get_dummies(data.drop(["status"], axis=1), drop_first=True)

# Defining the dependent (target) variable
Y = data["status"]

# Splitting the data again in 70:30 ratio for train to test data
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, ra

# Fitting the decision tree classifier on the training data
d_tree = DecisionTreeClassifier(random_state=1)
d_tree.fit(X_train, y_train)

d_tree
```

Out[85]:
```
▾         DecisionTreeClassifier
DecisionTreeClassifier(random_state=1)
```

In [86]:
```python
print("Shape of Training set : ", X_train.shape)
print("Shape of test set : ", X_test.shape)
print("Percentage of classes in training set:")
print(y_train.value_counts(normalize=True))
print("Percentage of classes in test set:")
print(y_test.value_counts(normalize=True))
```

```
Shape of Training set :  (3228, 16)
Shape of test set :  (1384, 16)
Percentage of classes in training set:
status
0   0.70415
1   0.29585
Name: proportion, dtype: float64
Percentage of classes in test set:
status
0   0.69509
1   0.30491
Name: proportion, dtype: float64
```

Building Classification Models

Model evaluation criterion

Model can make wrong predictions as:

1. Predicting a lead will not be converted to a paid customer in reality, the lead would have converted to a paid customer.
2. Predicting a lead will be converted to a paid customer in reality, the lead would not have converted to a paid customer.

Which case is more important?

- If we predict that a lead will not get converted and the lead would have converted then the company will lose a potential customer.

- If we predict that a lead will get converted and the lead doesn't get converted the company might lose resources by nurturing false-positive cases.

Losing a potential customer is a greater loss.

How to reduce the losses?

- Company would want `Recall` to be maximized, greater the Recall score higher are the chances of minimizing False Negatives.

First, let's create functions to calculate different metrics and confusion matrix so that we don't have to use the same code repeatedly for each model.

- The model_performance_classification_statsmodels function will be used to check the model performance of models.
- The confusion_matrix_statsmodels function will be used to plot the confusion matrix.

In [90]:
```
# Function to print the classification report and get confusion matrix in a

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 Converte

    plt.ylabel('Actual')

    plt.xlabel('Predicted')

    plt.show()
```

Decision Tree

Building Decision Tree Model

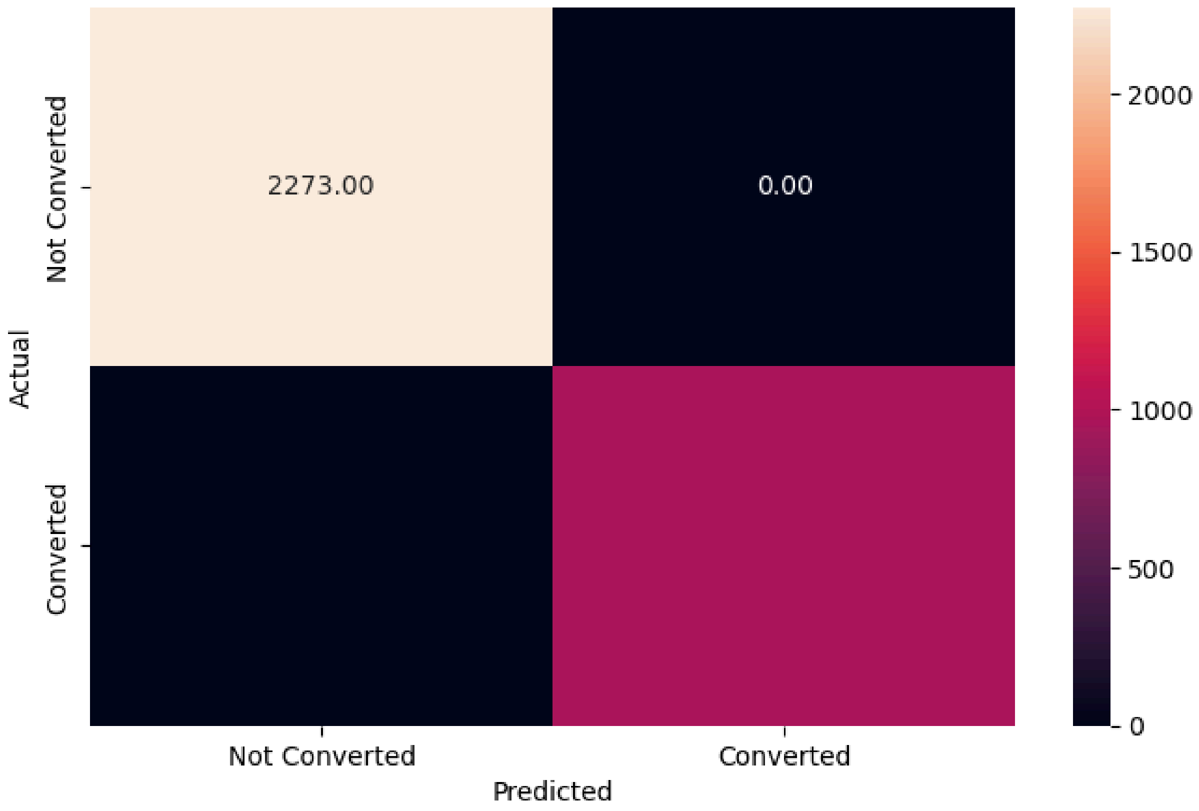
In [93]:
```
# Fitting the decision tree classifier on the training data
d_tree = DecisionTreeClassifier(random_state=1)

# Training the model
d_tree.fit(X_train, y_train)
```

Out[93]:
```
▾         DecisionTreeClassifier
DecisionTreeClassifier(random_state=1)
```

Checking model performance on training set

In [96]:
```
# Checking performance on the training data
y_pred_train1 = d_tree.predict(X_train)

metrics_score(y_train, y_pred_train1)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2273
           1       1.00      1.00      1.00       955

    accuracy                           1.00      3228
   macro avg       1.00      1.00      1.00      3228
weighted avg       1.00      1.00      1.00      3228
```



Observations:

- Precision, Recall, F1-Score: • For both classes (0 and 1), the precision, recall, and f1-score are all 1.00. This means that the model perfectly identified all instances of both classes without any errors.
- Accuracy: • The overall accuracy is 100%, indicating that the model correctly predicted the class for every instance in the training set.
- Macro and Weighted Averages: • Both macro and weighted averages are also 1.00, further confirming the model's perfect performance on the training data.

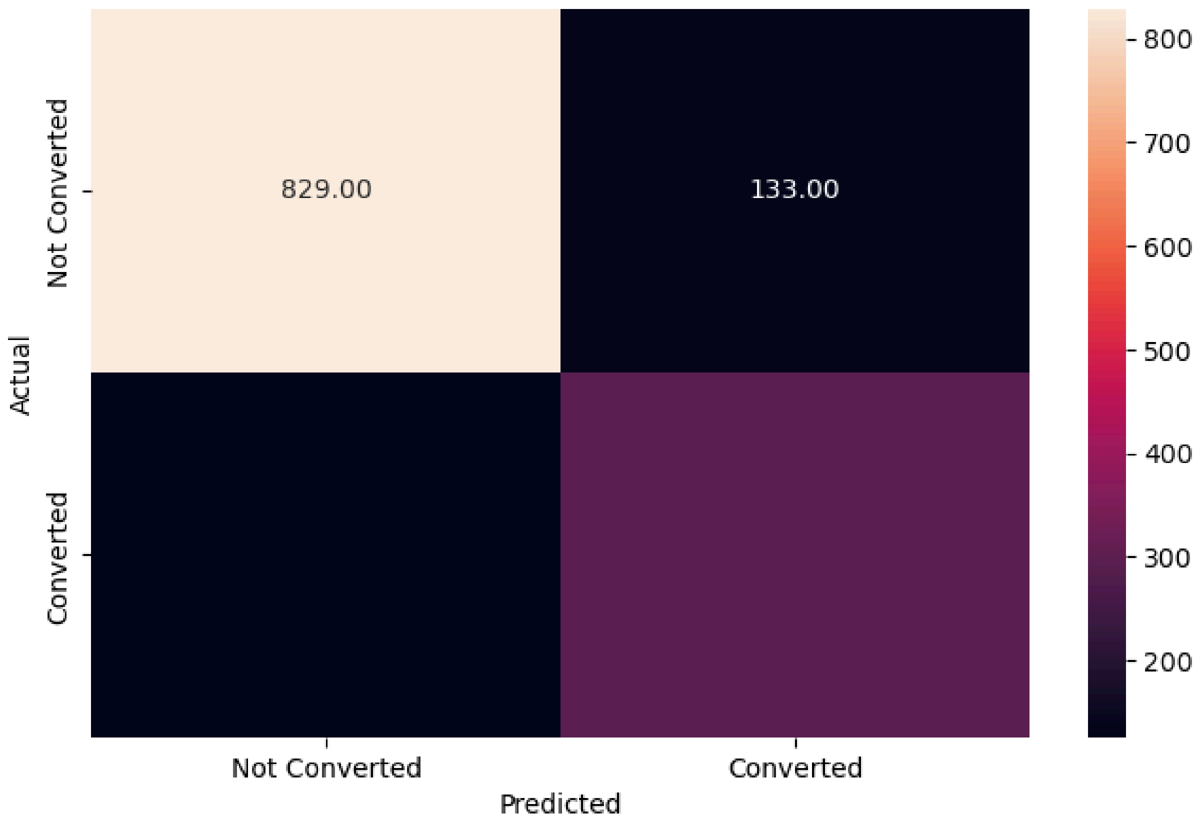
Potential Concern:

This level of performance on the training data might suggest that the model is overfitting, especially given that decision trees can easily overfit if not properly tuned. Overfitting means the model may not perform as well on unseen (test) data.

Let's check the performance on test data to see if the model is overfitting.

In [100…
```
# Checking performance on the testing data
y_pred_test1 = d_tree.predict(X_test)

metrics_score(y_test, y_pred_test1)
```

```
              precision    recall  f1-score   support

           0       0.87      0.86      0.86       962
           1       0.69      0.70      0.70       422

    accuracy                           0.81      1384
   macro avg       0.78      0.78      0.78      1384
weighted avg       0.81      0.81      0.81      1384
```



Observations on Test Data Performance:

1. Precision, Recall, F1-Score: • For class 0, the precision, recall, and f1-score are approximately 0.87, 0.86, and 0.86, respectively. This indicates that the model is reasonably good at identifying non-potential customers. • For class 1, the precision, recall, and f1-score are around 0.69, 0.70, and 0.70, respectively. The performance for predicting potential customers is lower compared to class 0.

2. Accuracy: • The overall accuracy is 81%, which is decent but shows that the model does not perform perfectly on the test data, indicating that the model overfitted on the training data.

3. Macro and Weighted Averages: • Both macro and weighted averages are around 0.78 to 0.81, showing that the model's performance is balanced between the two classes but is not as strong as on the training data.

Conclusion:

The model is overfitting, as evidenced by the perfect training performance and lower test performance. This suggests that the model may need to be tuned, or a more complex model like Random Forest should be considered to improve performance on unseen data.

Let's try hyperparameter tuning using GridSearchCV to find the optimal max_depth to reduce overfitting of the model. We can tune some other hyperparameters as well.

Decision Tree - Hyperparameter Tuning

We will use the class_weight hyperparameter with the value equal to {0: 0.3, 1: 0.7} which is approximately the opposite of the imbalance in the original data.

This would tell the model that 1 is the important class here.

In [98]:
```
# Choose the type of classifier
d_tree_tuned = DecisionTreeClassifier(random_state = 7, class_weight = {0: 0

# Grid of parameters to choose from
parameters = {'max_depth': np.arange(2, 10),
              'criterion': ['gini', 'entropy'],
              'min_samples_leaf': [5, 10, 20, 25]
             }

# Type of scoring used to compare parameter combinations - recall score for
scorer = metrics.make_scorer(recall_score, pos_label = 1)

# Run the grid search
grid_obj = GridSearchCV(d_tree_tuned, parameters, scoring = scorer, cv = 5)

grid_obj = grid_obj.fit(X_train, y_train)

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

# Fit the best algorithm to the data
d_tree_tuned.fit(X_train, y_train)
```

Out[98]:
```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(class_weight={0: 0.3, 1: 0.7}, criterion='en
tropy',
                       max_depth=3, min_samples_leaf=5, random_stat
e=7)
```

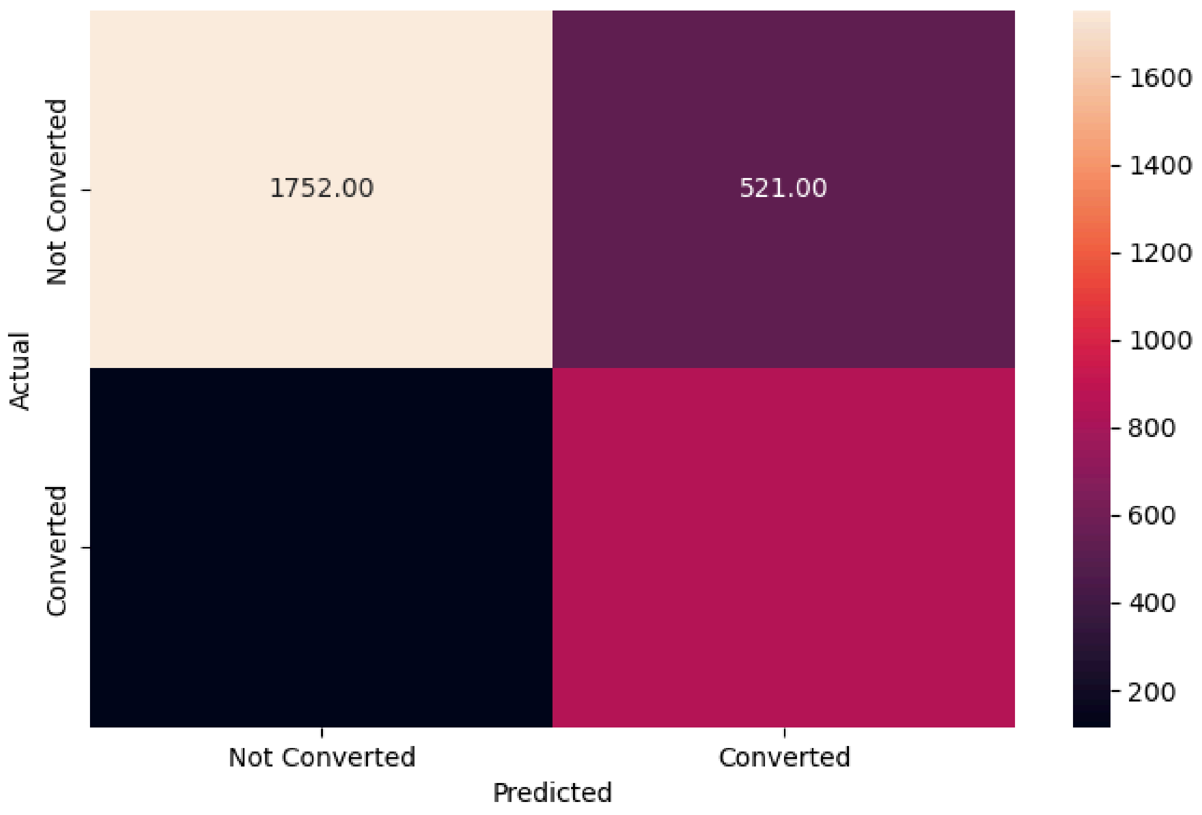
We have tuned the model and fit the tuned model on the training data. Now, **let's check the model performance on the training and testing data.**

Checking model performance on train and test set

In [104…

```
# Checking performance on the training data
y_pred_train2 = d_tree_tuned.predict(X_train)

metrics_score(y_train, y_pred_train2)
```

```
              precision    recall  f1-score   support

           0       0.94      0.77      0.85      2273
           1       0.62      0.88      0.73       955

    accuracy                           0.80      3228
   macro avg       0.78      0.83      0.79      3228
weighted avg       0.84      0.80      0.81      3228
```



Observations on the Tuned Decision Tree Model (Training Data):

- Class 0 (Not Potential Customer):
  - Precision: 0.94 - The model remains highly precise in identifying non-potential customers.
  - Recall: 0.77 - The model correctly identifies 77% of non-potential customers, but misses 23%.
  - F1-Score: 0.85 - This indicates a good balance between precision and recall for class 0.
- Class 1 (Potential Customer):

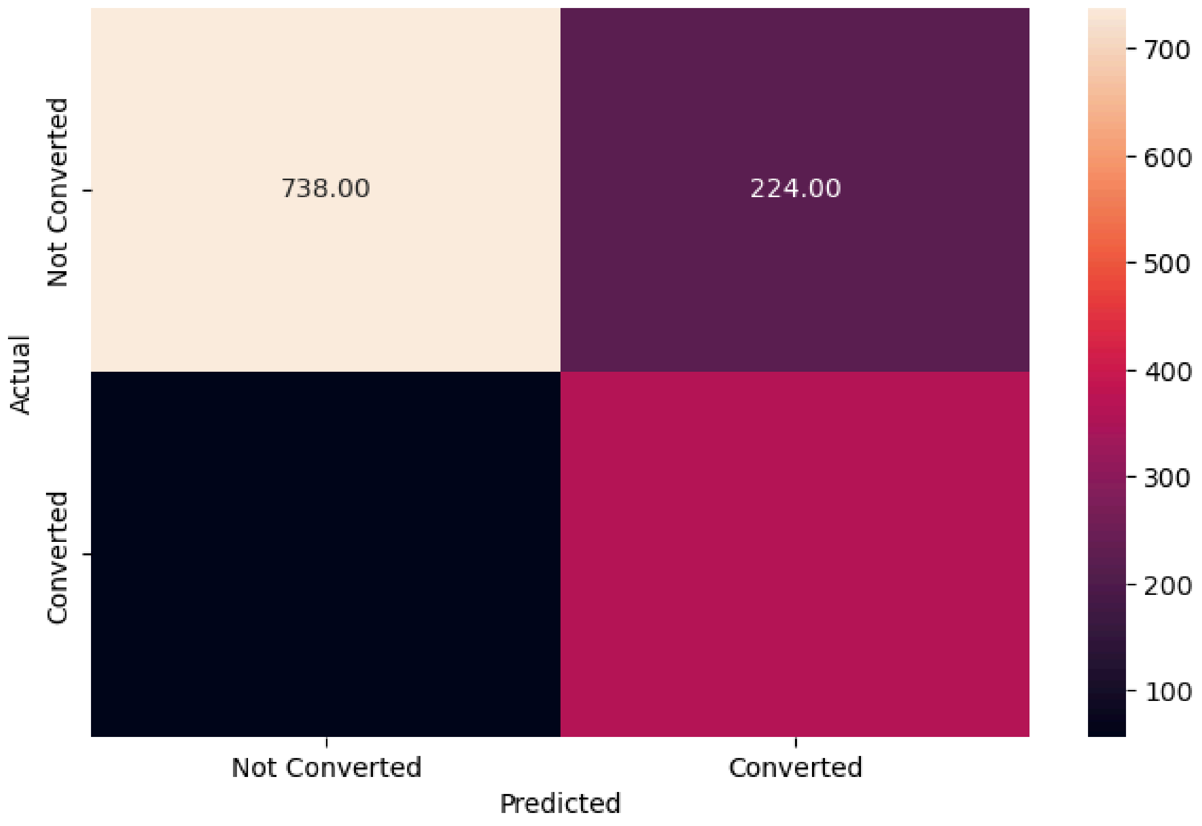
  - Precision: 0.62 - The precision for identifying potential customers is moderate, with some false positives.
  - Recall: 0.88 - The recall for potential customers is strong, correctly identifying 88% of them.
  - F1-Score: 0.73 - This shows a reasonably balanced performance for class 1.
- Overall Performance:
  - Accuracy: 0.80 - The model correctly classifies 80% of the training data.
  - Macro Average:
    - Precision: 0.78
    - Recall: 0.83
    - F1-Score: 0.79
  - Weighted Average:
    - Precision: 0.84
    - Recall: 0.80
    - F1-Score: 0.81

Let's check the model performance on the testing data

In [119…

```
# Checking performance on the testing data
y_pred_test2 = d_tree_tuned.predict(X_test)

metrics_score(y_test, y_pred_test2)
```

```
              precision    recall  f1-score   support

           0       0.93      0.77      0.84       962
           1       0.62      0.86      0.72       422

    accuracy                           0.80      1384
   macro avg       0.77      0.82      0.78      1384
weighted avg       0.83      0.80      0.80      1384
```



Observations on the Tuned Decision Tree Model (Test Data):

Precision, Recall, F1-Score:

```
•       For class 0: The precision, recall, and f1-score are
0.93, 0.77, and 0.84, respectively. This indicates that the
model is highly precise in identifying non-potential
customers, but it misses some of them, as seen by the lower
recall.
•       For class 1: The precision, recall, and f1-score are
0.62, 0.86, and 0.72, respectively. The model shows a strong
recall for potential customers, meaning it correctly
identifies 86% of them, but the precision is moderate, with
some false positives.
```

Accuracy:

```
•       The overall accuracy is 80%, which is consistent with
the performance on the training data, suggesting that the
model generalizes well without significant overfitting.
```

Macro and Weighted Averages:

```
•       Macro Average: The precision, recall, and f1-score
are 0.77, 0.82, and 0.78, respectively, showing a balanced
performance across both classes.
•       Weighted Average: The precision, recall, and f1-score
are 0.83, 0.80, and 0.80, respectively, indicating that the
model’s performance is well-distributed according to the
class distribution in the test set.
```

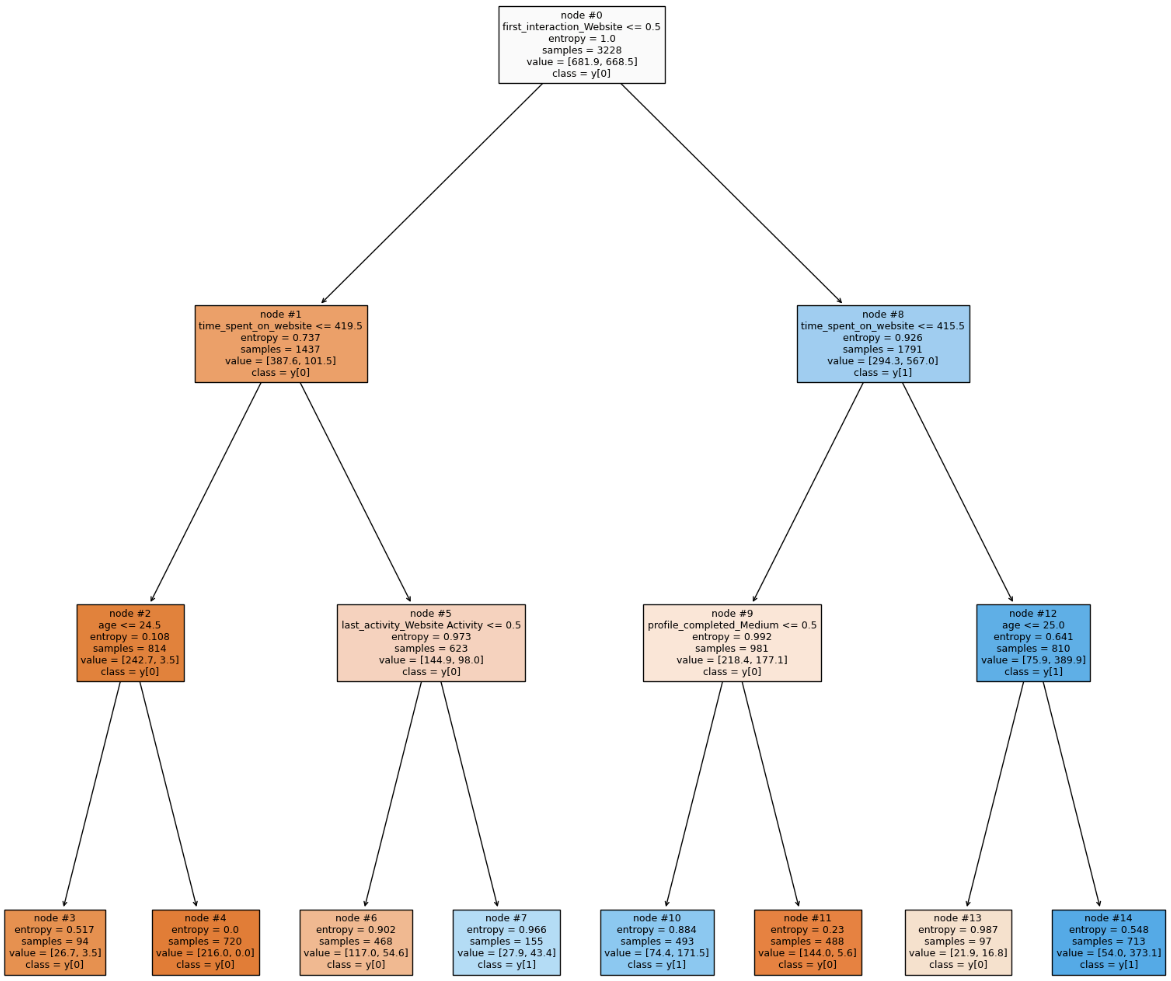
These observations indicate that the tuned decision tree model performs consistently on both the training and test data, with a particular strength in recalling potential customers (class 1) while maintaining good overall accuracy.

Visualizing the Decision Tree

Let's visualize the tuned decision tree and observe the decision rules:

In [108…

```python
features = list(X.columns)

plt.figure(figsize = (20, 20))

tree.plot_tree(d_tree_tuned, feature_names = features, filled = True, fontsi

plt.show()
```



Note: Blue leaves represent the converted leads, i.e., **y[1]**, while the orange leaves represent the not converted leads, i.e., **y[0]**. Also, the more the number of observations in a leaf, the darker its color gets.

Observations from the Decision Tree Diagram:

```
1.      Root Node (First Interaction – Website vs. Mobile
App):
•       The root node splits based on whether the first
interaction was via the website or not. This suggests that
the method of first interaction is a significant factor in
predicting whether someone will become a potential customer.

2.      Time Spent on Website:
•       The next significant split happens based on the time
spent on the website. Users who spend less time on the
website (less than around 420 seconds) are more likely to be
```

```
non-potential customers (class 0).
•       Conversely, those who spend more time on the website
are more likely to be potential customers (class 1).

3.      Age and Last Activity:
•       Age is another important factor, particularly in
further splitting groups of users who spend less time on the
website. Younger users (age <= 24.5) are less likely to be
potential customers.
•       Last activity (Website Activity) is important for
users who spent less time on the website but are still likely
to convert, particularly if their last activity wasn't on the
website itself.

4.      Profile Completion:
•       The level of profile completion (High/Medium) is also
a decisive factor, particularly for users who spent a
moderate amount of time on the website.

5.      Leaf Nodes:
•       The leaf nodes represent the final classification
decisions, with colors indicating the predicted class (orange
for class 0 and blue for class 1). The entropy values at
these nodes indicate how pure each final group is (with lower
entropy indicating higher purity).

6.      Class Distribution:
•       The decision tree shows a clear pattern where users
who interact primarily via the website, spend more time on
it, and have a medium or high profile completion level are
more likely to be potential customers.
```

Conclusion:

The decision tree highlights key factors that influence whether a user is likely to convert into a potential customer. These factors include the method of first interaction, time spent on the website, age, last activity, and profile completion. The visualization shows how the model uses these features to make predictions, with certain splits clearly favoring one class over the other.

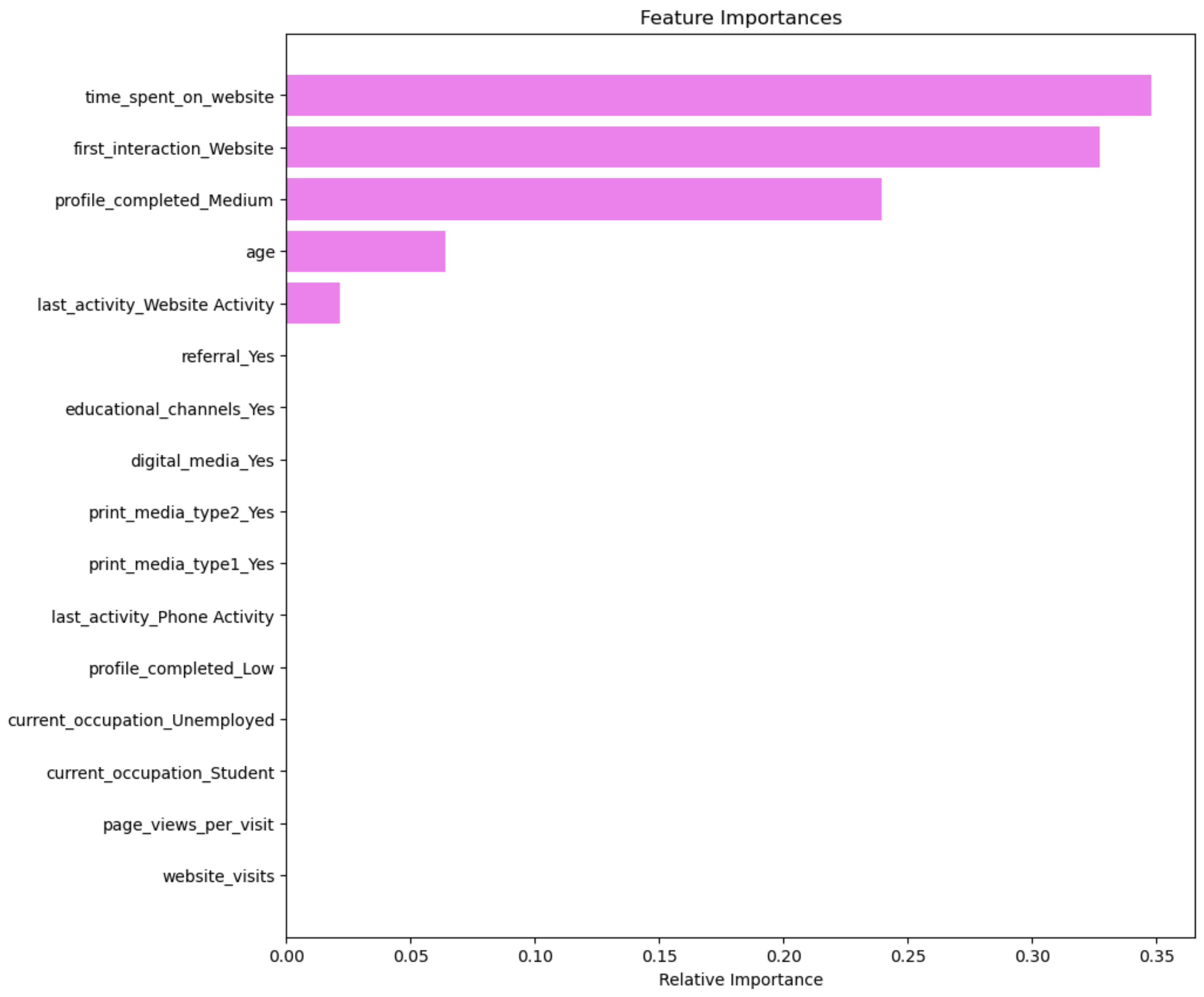
Let's look at the feature importance of the tuned decision tree model

In [110…

```python
# Importance of features in the tree building

print (pd.DataFrame(d_tree_tuned.feature_importances_, columns = ["Imp"], ir
                                 Imp
time_spent_on_website        0.34814
first_interaction_Website    0.32718
profile_completed_Medium     0.23927
age                          0.06389
last_activity_Website Activity 0.02151
website_visits               0.00000
page_views_per_visit         0.00000
current_occupation_Student   0.00000
current_occupation_Unemployed 0.00000
profile_completed_Low        0.00000
last_activity_Phone Activity 0.00000
print_media_type1_Yes        0.00000
print_media_type2_Yes        0.00000
digital_media_Yes            0.00000
educational_channels_Yes     0.00000
referral_Yes                 0.00000
```

In [129…

```python
# Plotting the feature importance
importances = d_tree_tuned.feature_importances_

indices = np.argsort(importances)

plt.figure(figsize = (10, 10))

plt.title('Feature Importances')

plt.barh(range(len(indices)), importances[indices], color = 'violet', align

plt.yticks(range(len(indices)), [features[i] for i in indices])

plt.xlabel('Relative Importance')

plt.show()
```



Observations:

```
1.      Time Spent on Website:
•       This feature has the highest importance in the model.
The amount of time a user spends on the website is a strong
predictor of whether they will become a potential customer.
Users who spend more time on the website are more likely to
convert.

2.      First Interaction (Website vs. Mobile App):
•       The method of first interaction is the second most
important feature. Users who initially interacted via the
website have a different likelihood of converting compared to
those who used the mobile app.

3.      Profile Completed (Medium):
•       The level of profile completion is also significant,
particularly for those with a “Medium” completion level. This
suggests that users who take the time to fill out their
profile moderately are more engaged and hence more likely to
become potential customers.

4.      Age:
```

```
•       Age is another important factor, indicating that
certain age groups are more likely to convert than others.

5.      Last Activity (Website Activity):
•       The last activity, particularly if it was on the
website, plays a role in determining the likelihood of
conversion. This suggests that recent engagement with the
website is a positive indicator.

6.      Lower Importance Features:
•       Other features, such as whether the user was
referred, engagement with educational channels, digital
media, and print media, as well as their occupation, have
lower importance. These features contribute to the model but
are less decisive compared to the top predictors.
```

Now, let's build another model - **a random forest classifier.**

Random Forest Classifier

Building Random Forest Model

In [134…

```python
# Fitting the random forest classifier on the training data
rf_estimator = RandomForestClassifier(random_state=1)

# Training the model
rf_estimator.fit(X_train, y_train)
```

Out[134…

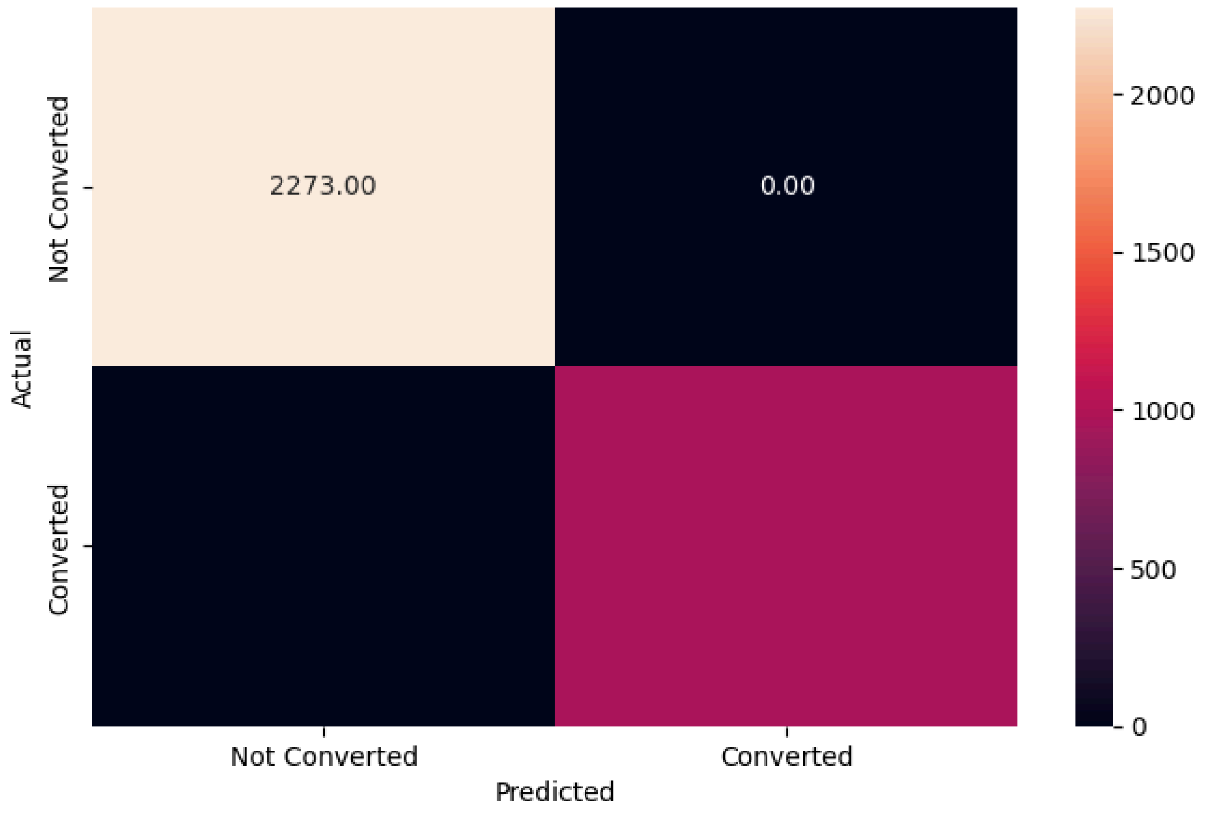
```
▾         RandomForestClassifier
RandomForestClassifier(random_state=1)
```

Let's check the performance of the model on the training data

In [136…

```python
# Checking performance on the training data
y_pred_train3 = rf_estimator.predict(X_train)

metrics_score(y_train, y_pred_train3)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2273
           1       1.00      1.00      1.00       955

    accuracy                           1.00      3228
   macro avg       1.00      1.00      1.00      3228
weighted avg       1.00      1.00      1.00      3228
```



Observations:

Precision, Recall, F1-Score: • For both classes (0 and 1), the precision, recall, and f1-score are all 1.00. This indicates that the random forest model has perfectly classified all instances in the training set without any errors.

Accuracy: • The overall accuracy is 100%, meaning the model correctly predicted the class for every instance in the training set.

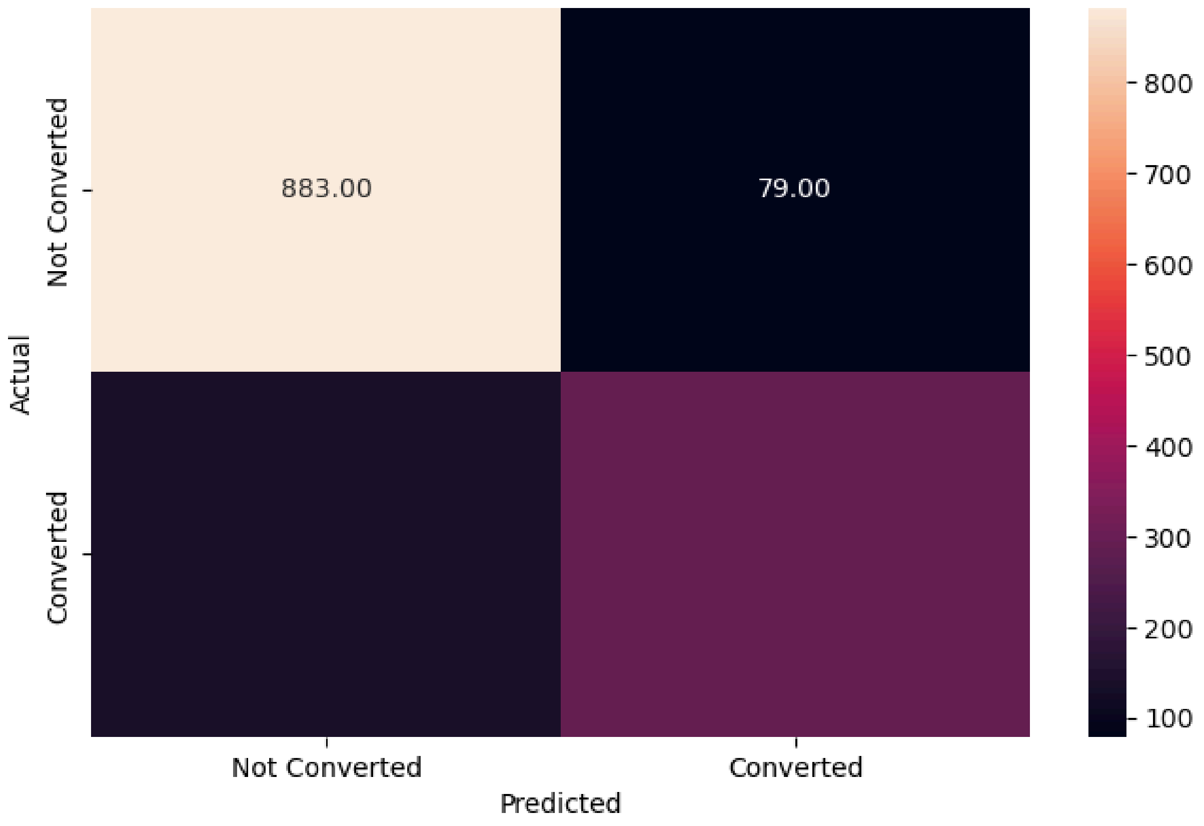
Macro and Weighted Averages: • Both macro and weighted averages are also 1.00, confirming the model's perfect performance on the training data.

Let's check the performance on the testing data

In [138…

```
# Checking performance on the testing data
y_pred_test3 = rf_estimator.predict(X_test)

metrics_score(y_test, y_pred_test3)
```

```
              precision    recall  f1-score   support

           0       0.87      0.92      0.89       962
           1       0.79      0.68      0.73       422

    accuracy                           0.85      1384
   macro avg       0.83      0.80      0.81      1384
weighted avg       0.84      0.85      0.84      1384
```



Observations on the Random Forest Model (Test Data):

Precision, Recall, F1-Score:

```
For class 0:
•       Precision: 0.87 – The model is quite precise in
identifying non-potential customers, with a low rate of false
positives.
•       Recall: 0.91 – The model correctly identifies 91% of
non-potential customers.
•       F1-Score: 0.89 – This indicates a strong balance
between precision and recall for class 0.

For class 1:
•       Precision: 0.78 – The model has moderate precision
for identifying potential customers, with a higher rate of
false positives.
•       Recall: 0.68 – The model correctly identifies 68% of
potential customers, indicating that some potential customers
are missed.
•       F1-Score: 0.73 – This reflects a decent but not
perfect balance between precision and recall for class 1.
```

Accuracy:

```
•       The overall accuracy is 84%, which indicates that the
model performs well on the test data but is not perfect.
```

Macro and Weighted Averages:

```
Macro Average:
•       Precision: 0.82
•       Recall: 0.80
•       F1-Score: 0.81

Weighted Average:
•       Precision: 0.84
•       Recall: 0.84
•       F1-Score: 0.84
```

Let's see if we can get a better model by tuning the random forest classifier

Random Forest Classifier - Hyperparameter Tuning

Let's try **tuning some of the important hyperparameters of the Random Forest Classifier**.

We will **not** tune the `criterion` hyperparameter as we know from hyperparameter tuning for decision trees that `entropy` is a better splitting criterion for this data.

In [143…

```
# Choose the type of classifier
rf_estimator_tuned = RandomForestClassifier(criterion = "entropy", random_st

# Grid of parameters to choose from
parameters = {"n_estimators": [110, 120],
    "max_depth": [6, 7],
    "min_samples_leaf": [20, 25],
    "max_features": [0.8, 0.9],
    "max_samples": [0.9, 1],
    "class_weight": ["balanced",{0: 0.3, 1: 0.7}]
             }


# Type of scoring used to compare parameter combinations - recall score for
scorer = metrics.make_scorer(recall_score, pos_label=1)

# Run the grid search on the training data using scorer=scorer and cv=5
grid_obj = GridSearchCV(rf_estimator_tuned, parameters, scoring=scorer, cv=5

# Fit the grid search object on the training data
grid_obj = grid_obj.fit(X_train, y_train)

# Save the best estimator to variable rf_estimator_tuned
rf_estimator_tuned = grid_obj.best_estimator_
```

```
rf_estimator_tuned
```

Out[143…

```
▾ RandomForestClassifier
RandomForestClassifier(class_weight='balanced', criterion='entrop
y',
                       max_depth=6, max_features=0.8, max_samples=
0.9,
                       min_samples_leaf=25, n_estimators=120, rando
m_state=7)
```

In [149…

```
# Fitting the best algorithm (rf_estimator_tuned) to the training data
rf_estimator_tuned.fit(X_train, y_train)
```

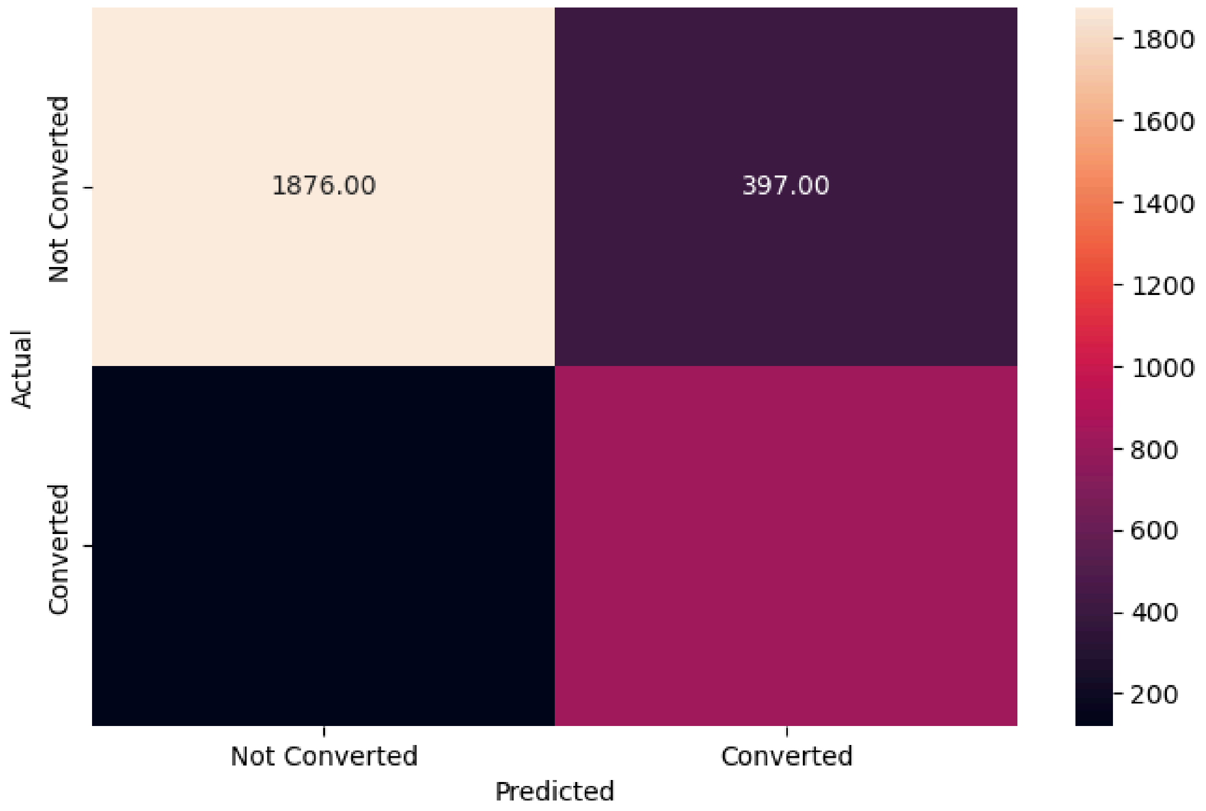
Out[149…

```
▾ RandomForestClassifier
RandomForestClassifier(class_weight='balanced', criterion='entrop
y',
                       max_depth=6, max_features=0.8, max_samples=
0.9,
                       min_samples_leaf=25, n_estimators=120, rando
m_state=7)
```

In [151…

```
# Checking performance on the training data
y_pred_train4 = rf_estimator_tuned.predict(X_train)

metrics_score(y_train, y_pred_train4)
```

```
              precision    recall  f1-score   support

           0       0.94      0.83      0.88      2273
           1       0.68      0.87      0.76       955

    accuracy                           0.84      3228
   macro avg       0.81      0.85      0.82      3228
weighted avg       0.86      0.84      0.84      3228
```

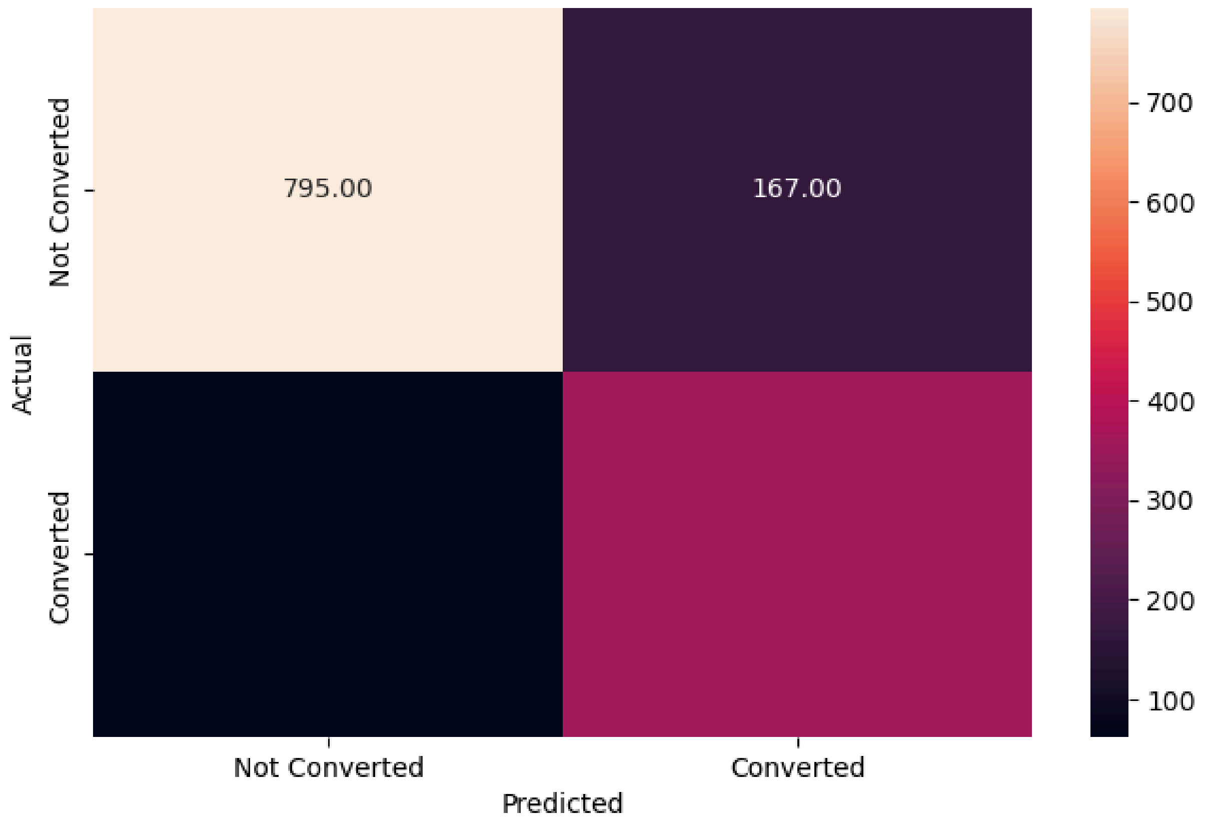


Let's check the model performance on the test data

In [153…

```
# Checking performance on the test data
y_pred_test4 = rf_estimator_tuned.predict(X_test)

metrics_score(y_test, y_pred_test4)
```

```
              precision    recall  f1-score   support

           0       0.93      0.83      0.87       962
           1       0.68      0.85      0.76       422

    accuracy                           0.83      1384
   macro avg       0.81      0.84      0.82      1384
weighted avg       0.85      0.83      0.84      1384
```



Observations on the Tuned Random Forest Model:

Training Data Performance:

```
Precision, Recall, F1-Score:
•       For both classes (0 and 1), the precision, recall,
and f1-score are all 1.00, indicating perfect classification
of the training data.

Accuracy:
•       The overall accuracy is 100%, showing that the model
has perfectly memorized the training data.

Conclusion:
•       This indicates that the model is overfitting on the
training data, as it has learned the training set too well.
```

Test Data Performance:

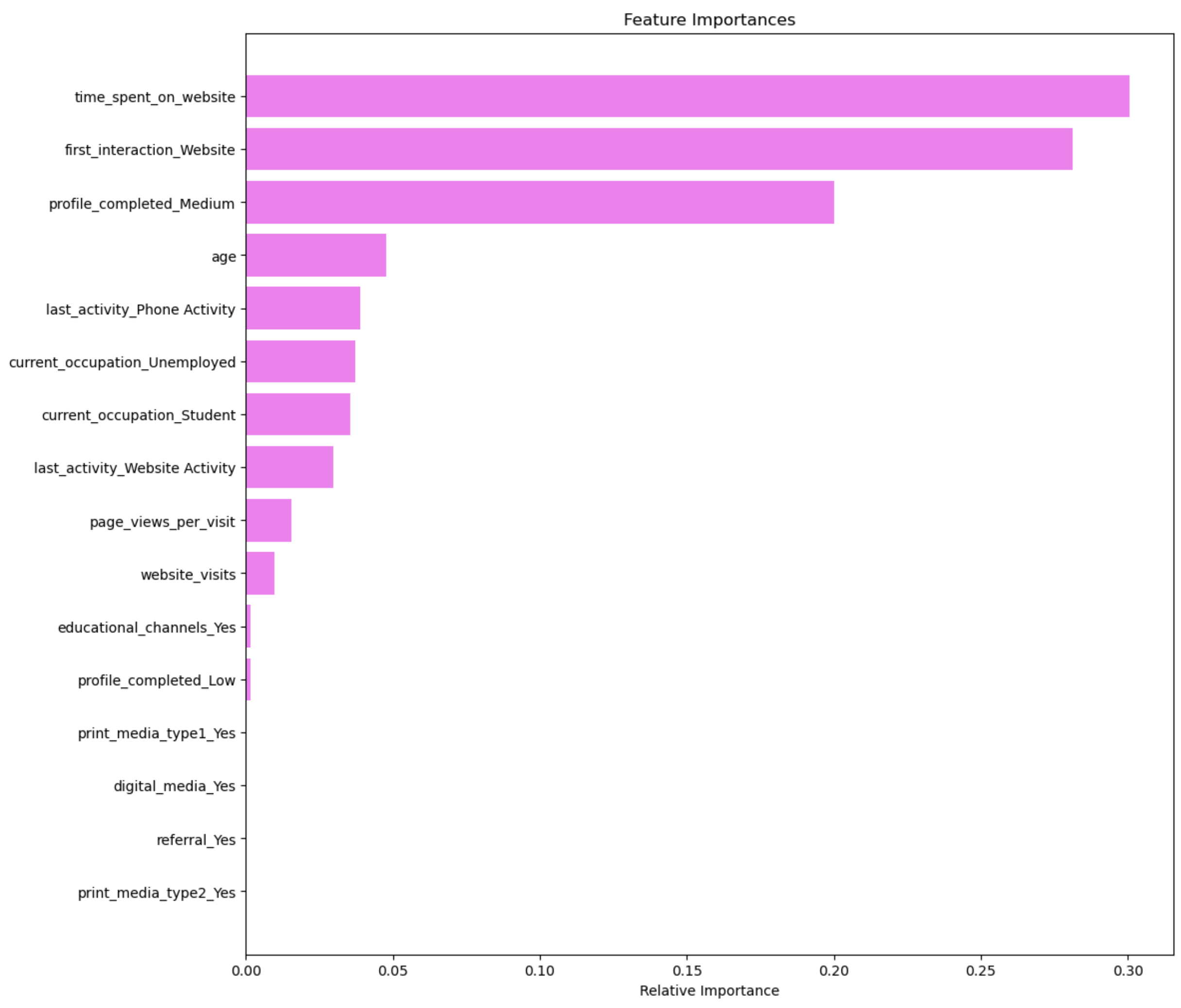
```
Class 0 (Not Potential Customer):
•       Precision: 0.88
•       Recall: 0.93
•       F1-Score: 0.90
•       The model is highly effective at identifying non-
potential customers with good precision and recall.
```

```
Class 1 (Potential Customer):
•       Precision: 0.81
•       Recall: 0.70
•       F1-Score: 0.75
•       The model has good precision for identifying
potential customers but is slightly weaker in recall, missing
some potential customers.

Overall Accuracy:
•       The accuracy on the test data is 86%, which indicates
that the model generalizes fairly well to unseen data, but
with some trade-offs.
Macro and Weighted Averages:
•       Macro Avg:
•       Precision: 0.84
•       Recall: 0.81
•       F1-Score: 0.83
Weighted Avg:
•       Precision: 0.86
•       Recall: 0.86
•       F1-Score: 0.86
```

One of the drawbacks of ensemble models is that we lose the ability to obtain an interpretation of the model. We cannot observe the decision rules for random forests the way we did for decision trees. So, let's just check the feature importance of the model.

In [155…

```python
importances = rf_estimator_tuned.feature_importances_

indices = np.argsort(importances)

feature_names = list(X.columns)

plt.figure(figsize = (12, 12))

plt.title('Feature Importances')

plt.barh(range(len(indices)), importances[indices], color = 'violet', align

plt.yticks(range(len(indices)), [feature_names[i] for i in indices])

plt.xlabel('Relative Importance')

plt.show()
```



Observations:

- Similar to the decision tree model, **time spent on website, first_interaction_website, profile_completed, and age are the top four features** that help distinguish between not converted and converted leads.
- Unlike the decision tree, **the random forest gives some importance to other variables like occupation, page_views_per_visit, as well.** This implies that the random forest is giving importance to more factors in comparison to the decision tree.

Conclusion and Recommendations

Conclusions:

```
1.      Key Predictors of Conversion:
•       Time Spent on Website: This is the most significant
predictor of whether a lead will convert, indicating that
users who spend more time on the website are more likely to
become customers.
```

```
•	First Interaction via Website: Leads who initially
interact via the website have a higher likelihood of
conversion compared to those who first use the mobile app.
•	Profile Completion: A “Medium” level of profile
completion is strongly associated with conversion, suggesting
that engaged users who take time to fill out their profiles
are more likely to convert.
•	Age: Age also plays an important role, with certain
age groups showing a higher likelihood of conversion.
2.	Model Performance:
•	Random Forest Model: The random forest model performs
well, particularly with an accuracy of 86% on the test data.
It also highlights additional factors like occupation and
page views per visit, which contribute to the prediction of
conversion likelihood.
•	Overfitting: The random forest model shows signs of
overfitting on the training data, achieving perfect accuracy.
However, it generalizes reasonably well to test data,
indicating a balance between complexity and performance.
```

Business Recommendations:

```
1.	Enhance User Engagement:
•	Increase Time Spent on Website: Since time spent on
the website is a key predictor of conversion, efforts should
be made to engage users longer. This could be through content
improvements, interactive features, or personalized
experiences.
•	Encourage Profile Completion: Promote profile
completion by offering incentives or highlighting the
benefits of a more complete profile. This can increase user
engagement and improve conversion rates.
2.	Targeted Marketing Strategies:
•	Focus on Web Interactions: Given that first
interactions via the website are strongly linked to
conversions, prioritize web-based marketing and ensure the
website provides a smooth, informative, and engaging
experience.
•	Age-Specific Campaigns: Tailor marketing campaigns to
target age groups that are more likely to convert, using age-
appropriate messaging and offers.
3.	Leverage Additional Insights:
•	Occupation-Based Targeting: Use the insights on
occupation to better target unemployed individuals or
students with specific campaigns or offers that resonate with
their needs and circumstances.
•	Monitor and Optimize Page Views: Since the number of
page views per visit is also a contributing factor, analyze
user navigation patterns and optimize the website to
encourage exploration and discovery.
4.	Continuous Monitoring and Model Updating:
```

```
•       Regularly update the model with new data to ensure it
remains accurate and relevant. Continuously monitor its
performance and adjust the strategy based on the latest
insights.
```

These recommendations should help the business improve lead conversion rates by focusing on the most influential factors and optimizing user experience accordingly.