Optimizing EdTech Leads Project

Author: Spencer Rogovin

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. This project utilizes a variety of Machine Learning models and exploratory data analysis techniques to help identify which leads are more likely to convert to paid customers, and the factors driving the lead conversion process.

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.

Importing necessary libraries and data

In [1]:
```
import warnings

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

warnings.simplefilter("ignore", ConvergenceWarning)

import pandas as pd
import numpy as np

# Split data library
from sklearn.model_selection import train_test_split
```

```
# data visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns

# limit for displayed columns removed
pd.set_option("display.max_columns", None)
# limit for displayed rows set
pd.set_option("display.max_rows", 200)
# precision of floating numbers set 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

# Tuning tools
from sklearn.model_selection import GridSearchCV


# To get metrics
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,
)
```

In [2]:
```
## Loading dataset
learn = pd.read_csv("ExtraaLearn.csv")
```

In [3]:
```
# copy of data to another variable to avoid changes to original data
data = learn.copy()
```

Data Overview

- Observations
- Sanity checks

In [4]:
```
## View first & last 5 rows of dataset
print(data.head())
print(data.tail())
```

```
       ID  age current_occupation first_interaction profile_completed  \
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

   website_visits  time_spent_on_website  page_views_per_visit  \
0               7                   1639               1.86100
1               2                     83               0.32000
2               3                    330               0.07400
3               4                    464               2.05700
4               4                    600              16.91400

     last_activity print_media_type1 print_media_type2 digital_media  \
0  Website Activity               Yes                No           Yes
1  Website Activity                No                No            No
2  Website Activity                No                No           Yes
3  Website Activity                No                No            No
4    Email Activity                No                No            No

  educational_channels referral  status
0                   No       No       1
1                  Yes       No       0
2                   No       No       0
3                   No       No       1
4                   No       No       0
           ID  age current_occupation first_interaction profile_completed  \
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

      website_visits  time_spent_on_website  page_views_per_visit  \
4607              15                    360               2.17000
4608               8                   2327               5.39300
4609               2                    212               2.69200
4610               1                    154               3.87900
4611               4                   2290               2.07500

         last_activity print_media_type1 print_media_type2 digital_media  \
4607    Phone Activity                No                No            No
4608    Email Activity                No                No            No
4609    Email Activity                No                No            No
4610  Website Activity               Yes                No            No
4611    Phone Activity                No                No            No

     educational_channels referral  status
4607                  Yes       No       0
4608                   No       No       0
4609                   No       No       1
4610                   No       No       0
4611                   No       No       0
```

In [12]:
```
print(data.shape) ## shape of dataset
print(data.dtypes) ## data types for columns
print(data.duplicated().sum()) ## duplicate value check
```

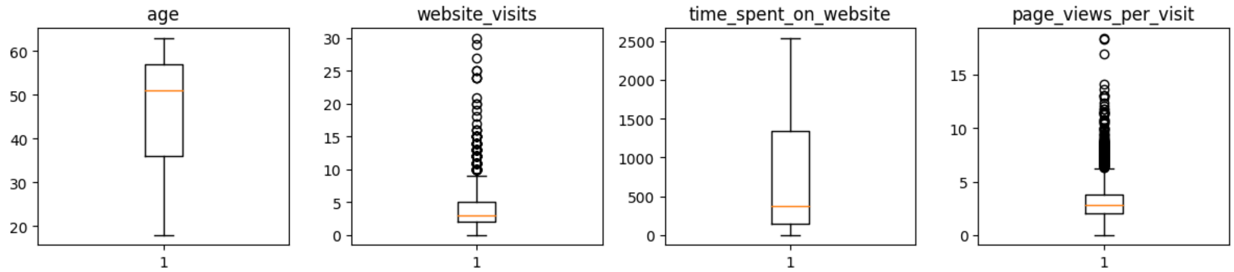
```
(4612, 15)
ID                         object
age                         int64
current_occupation         object
first_interaction          object
profile_completed          object
website_visits              int64
time_spent_on_website       int64
page_views_per_visit      float64
last_activity              object
print_media_type1          object
print_media_type2          object
digital_media              object
educational_channels       object
referral                   object
status                      int64
dtype: object
0
```

Data Preprocessing & Exploratory Data Analysis (EDA)

In [5]:
```
# import Data Preprocessing libraries
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
```

In [6]:
```
# renove irrelevant columns or columns with too many missing values
edtech_leads = data.drop(columns=['ID', 'last_activity'])

# Separate features and target variable
X = data.drop(columns=['status'])
y = data['status']

# Encode categorical variables
categorical_features = ['current_occupation', 'first_interaction', 'profile_co
numerical_features = ['age', 'website_visits', 'time_spent_on_website', 'page_
binary_features = ['print_media_type1', 'print_media_type2', 'digital_media',
```

In [7]:
```
# OUTLIER CHECK USIN
numeric_columns = edtech_leads.select_dtypes(include=np.number).columns.tolist
# drop release_year because it is a time-based variable --> n/a to outlier che
numeric_columns.remove("status")


plt.figure(figsize=(12, 10))

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()
```



In [8]:
```
edtech_leads.describe() ##  statistical summary of the data
```

Out[8]:

	age	website_visits	time_spent_on_website	page_views_per_visit	status
count	4612.00000	4612.00000	4612.00000	4612.00000	4612.00000
mean	46.20121	3.56678	724.01127	3.02613	0.29857
std	13.16145	2.82913	743.82868	1.96812	0.45768
min	18.00000	0.00000	0.00000	0.00000	0.00000
25%	36.00000	2.00000	148.75000	2.07775	0.00000
50%	51.00000	3.00000	376.00000	2.79200	0.00000
75%	57.00000	5.00000	1336.75000	3.75625	1.00000
max	63.00000	30.00000	2537.00000	18.43400	1.00000

In [9]:
```
# unique values in the "ID" column
print("Number of unique values in 'ID' column:", data["ID"].nunique())
```

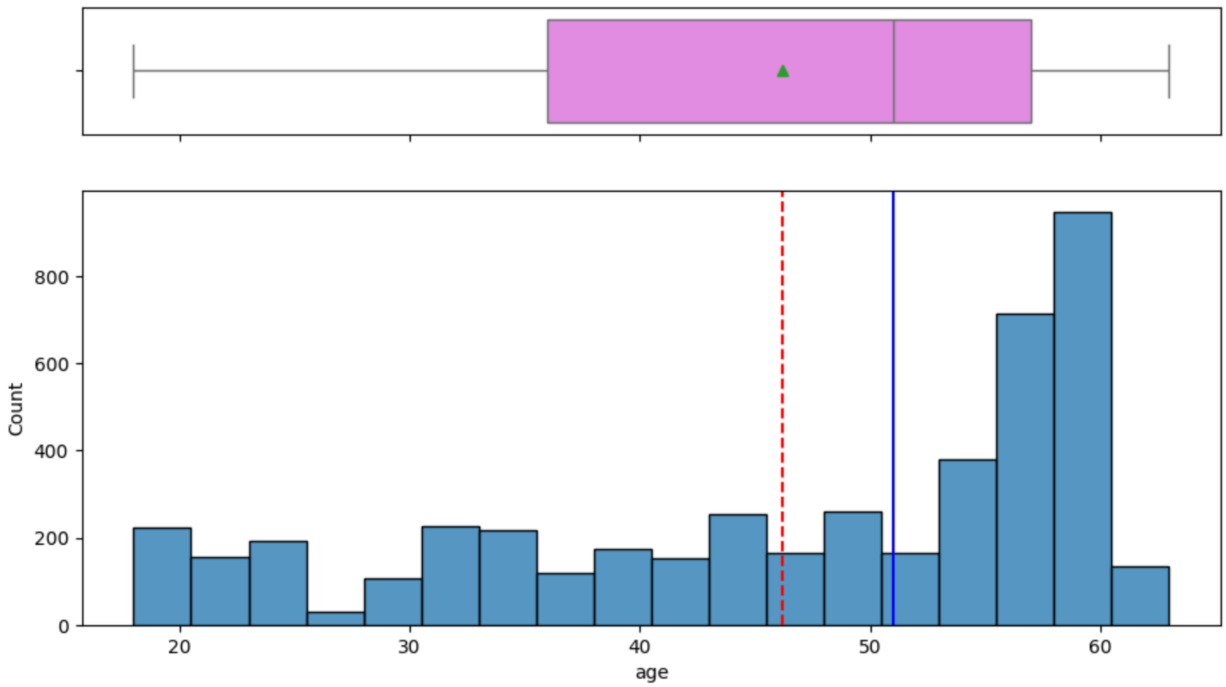
```
Number of unique values in 'ID' column: 4612
```

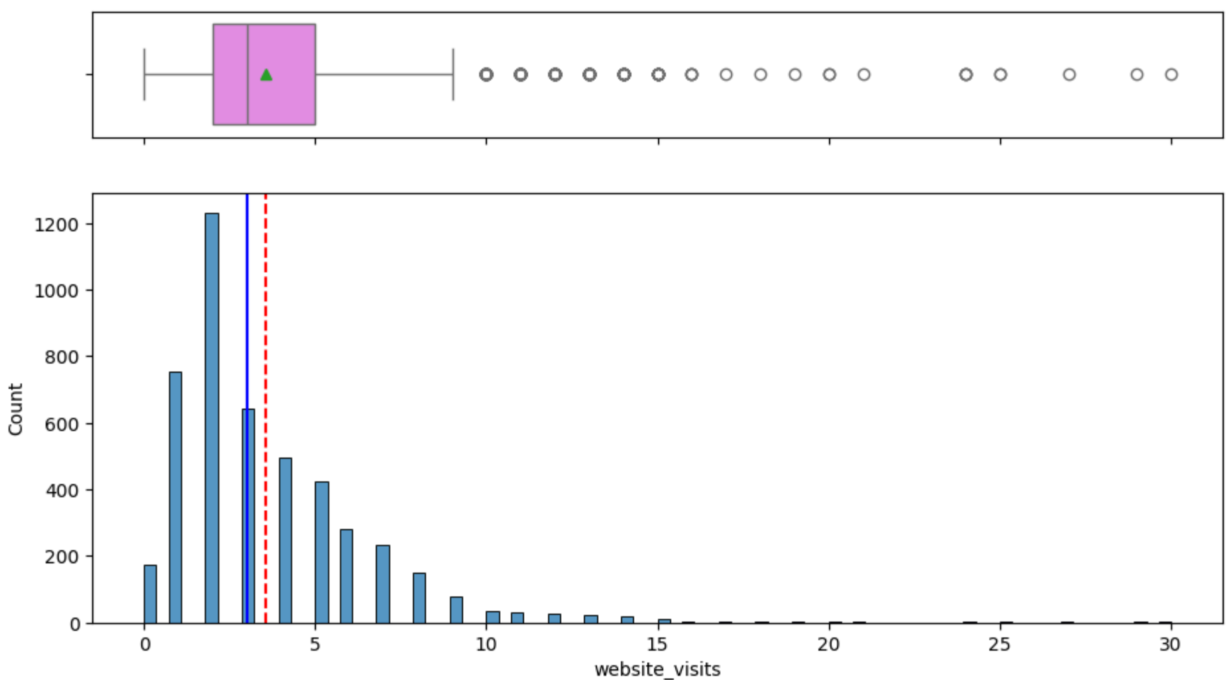
In [10]:
```
## SETTING UP UNIVARIATE ANALYSIS OF NUMERICAL FEATURES
# function to plot a boxplot and a histogram along the same scale.


def histogram_boxplot(edtech_leads, feature, figsize=(11, 6), kde=False, bins=N
    """
    Boxplot and histogram combined

    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (11,6))
    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 shared amongst all subplots
        gridspec_kw={"height_ratios": (0.20, 0.68)},
        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 the
    sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2, bins=bins, palette="summer"
    ) if bins else sns.histplot(
```

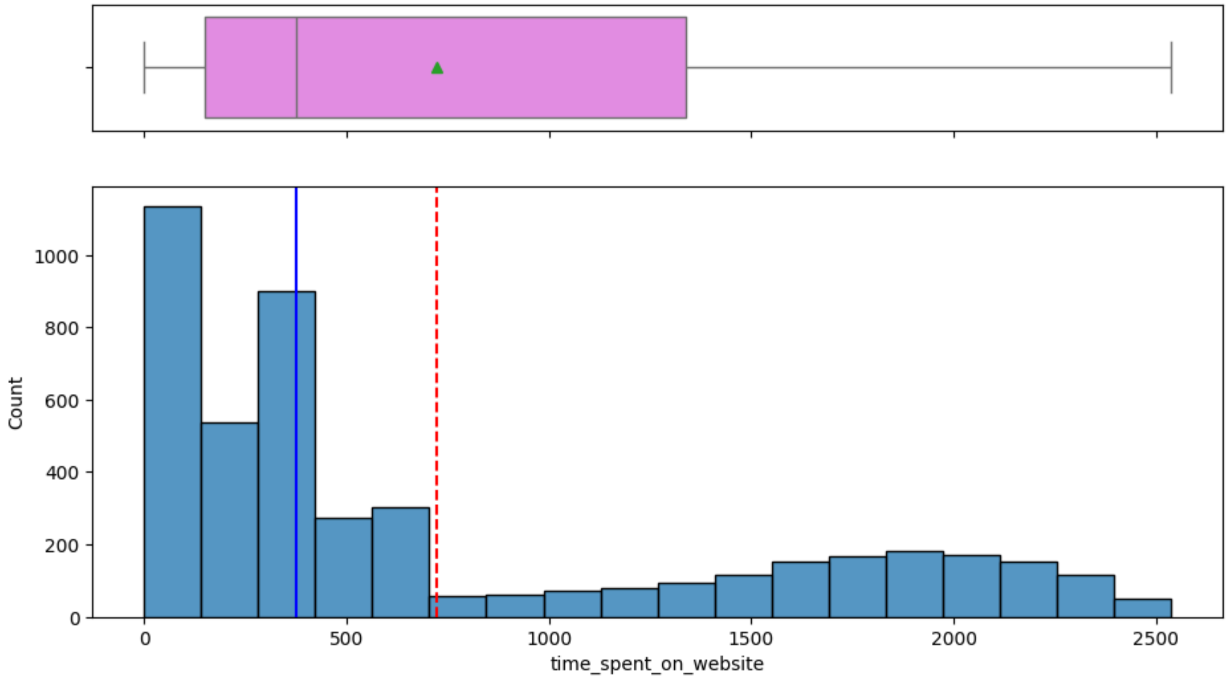
```
        data=data, x=feature, kde=kde, ax=ax_hist2
    )  # For histogram
    ax_hist2.axvline(
        data[feature].mean(), color="red", linestyle="--"
    )  # Add mean to the histogram
    ax_hist2.axvline(
        data[feature].median(), color="blue", linestyle="-"
    )  # Add median to the histogram
```

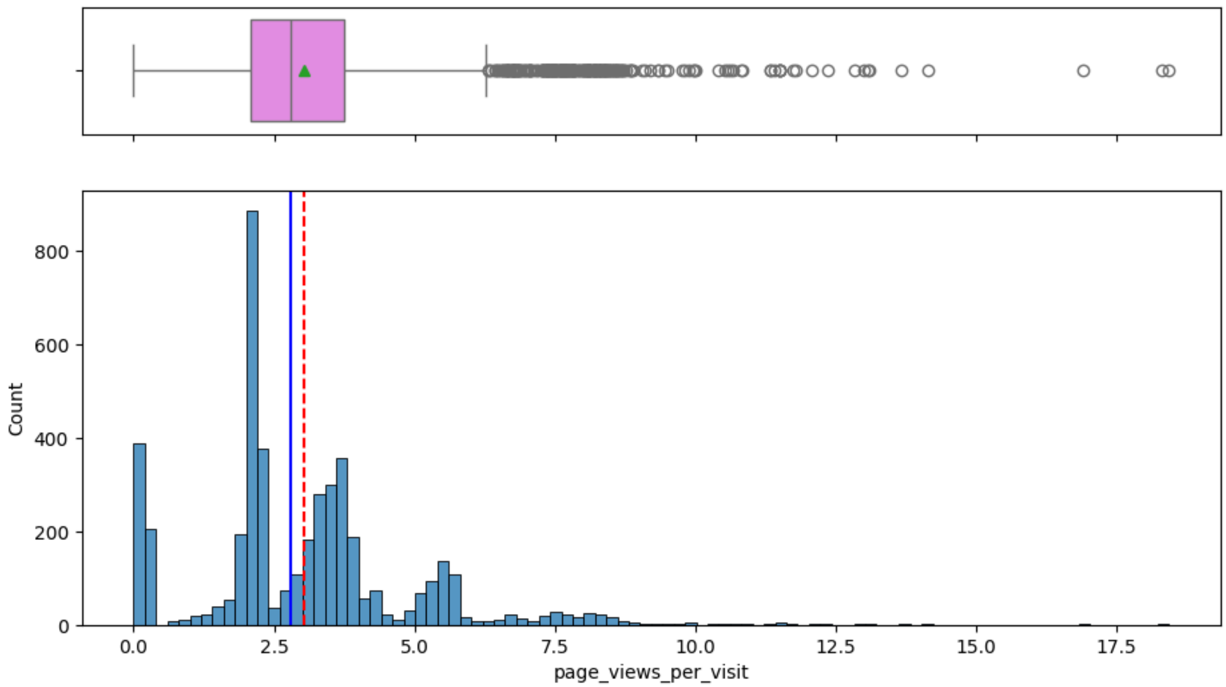
In [11]:

```
## UNIVARIATE ANALYSIS OF NUMERICAL FEATURES
histogram_boxplot(edtech_leads, "age")
histogram_boxplot(edtech_leads, "website_visits")
histogram_boxplot(edtech_leads, "time_spent_on_website")
histogram_boxplot(edtech_leads, "page_views_per_visit")
```





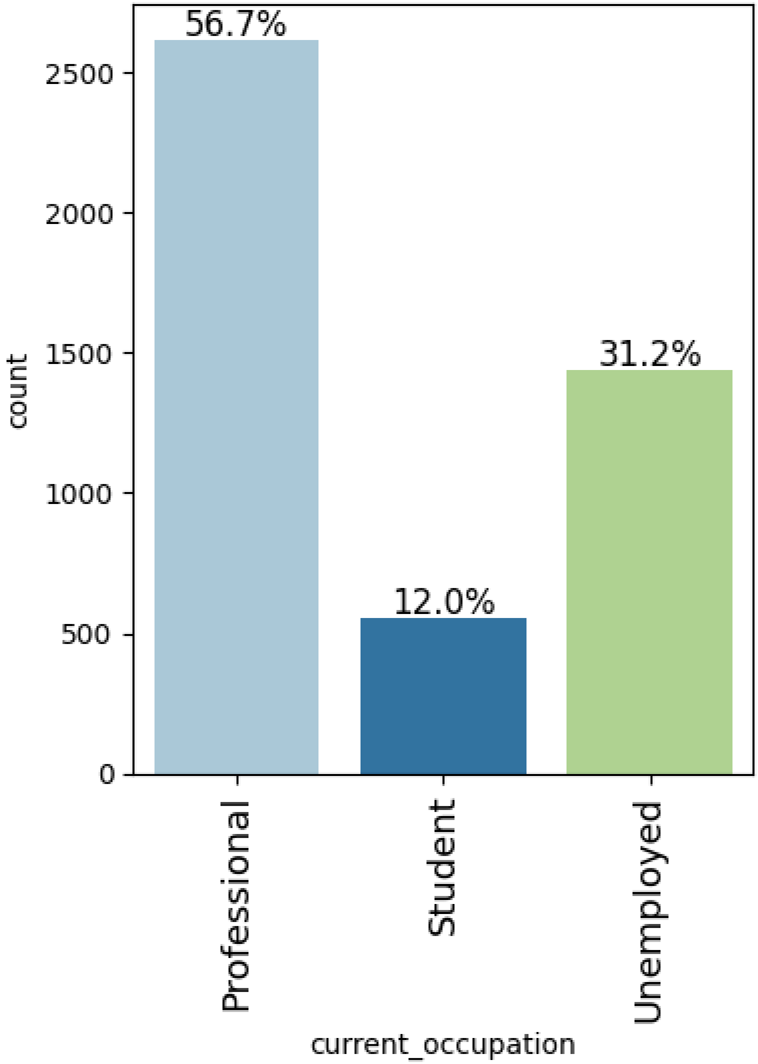


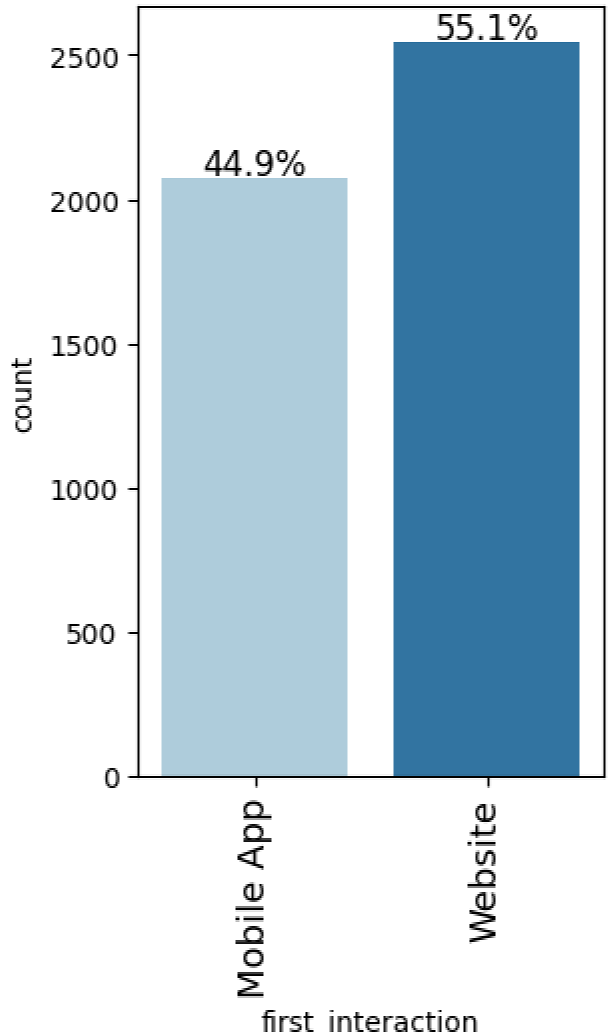


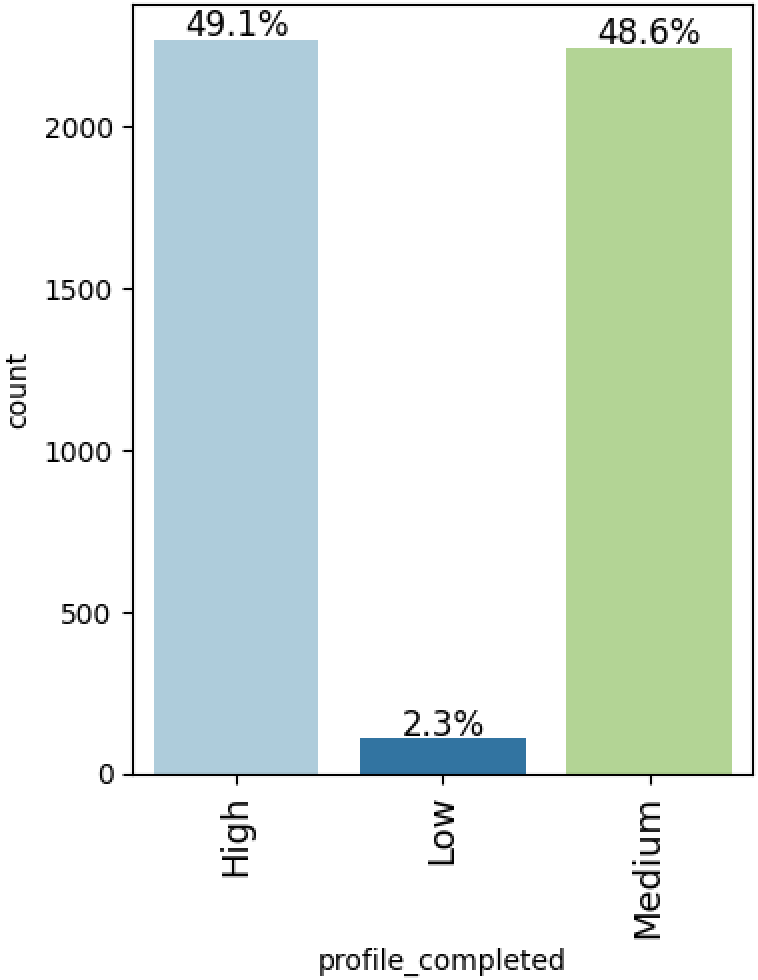
In [13]:
```
## SETTING UP UNIVARIATE ANALYSIS OF CATEGORICAL & BINARY FEATURES

def labeled_barplot(edtech_leads, 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 all
    """

    total = len(edtech_leads[feature])  # column length
    count = edtech_leads[feature].nunique()
    if n is None:
```

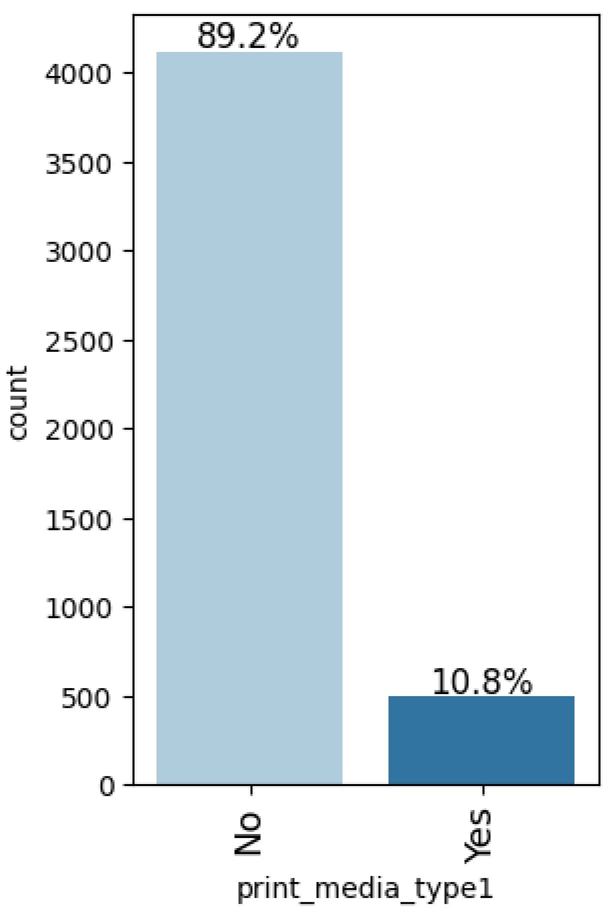
```
        plt.figure(figsize=(count + 1, 5))
    else:
        plt.figure(figsize=(n + 1, 5))

    plt.xticks(rotation=90, fontsize=13)
    ax = sns.countplot(
        data=edtech_leads,
        x=feature,
        palette="Paired",
        order=edtech_leads[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",
        )  # annotates the percentage

    plt.show()  # shows plot
```

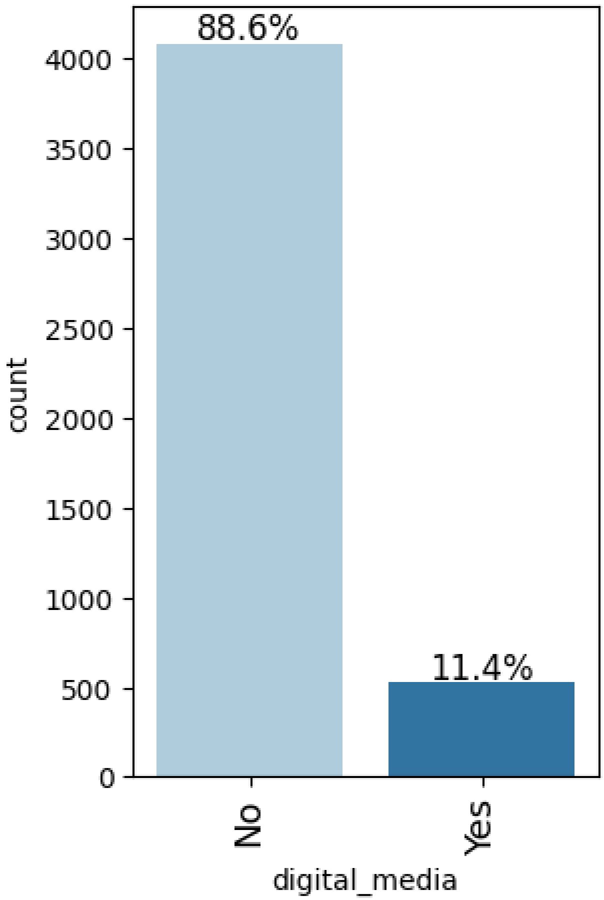
In [14]:
```
## UNIVARIATE ANALYSIS OF CATEGORICAL & BINARY FEATURES
labeled_barplot(edtech_leads, "current_occupation", perc=True)
labeled_barplot(edtech_leads, "first_interaction", perc=True)
labeled_barplot(edtech_leads, "profile_completed", perc=True)
labeled_barplot(edtech_leads, "print_media_type1", perc=True)
labeled_barplot(edtech_leads, "digital_media", perc=True)
labeled_barplot(edtech_leads, "educational_channels", perc=True)
labeled_barplot(edtech_leads, "referral", perc=True)
labeled_barplot(edtech_leads, "status", perc=True)
```

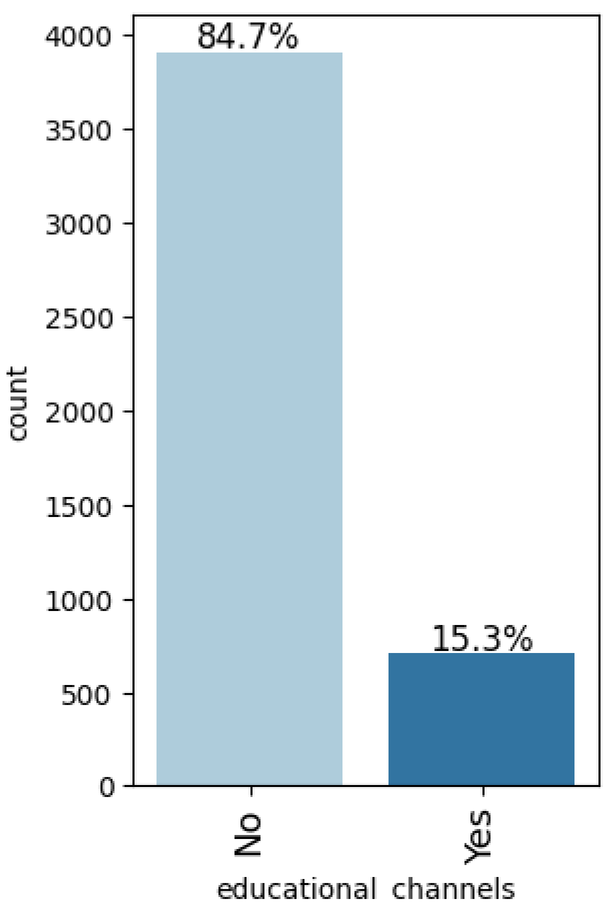

56.7%
12.0%
31.2%
count
2500
2000
1500
1000
500
0
Professional
Student
Unemployed
current_occupation

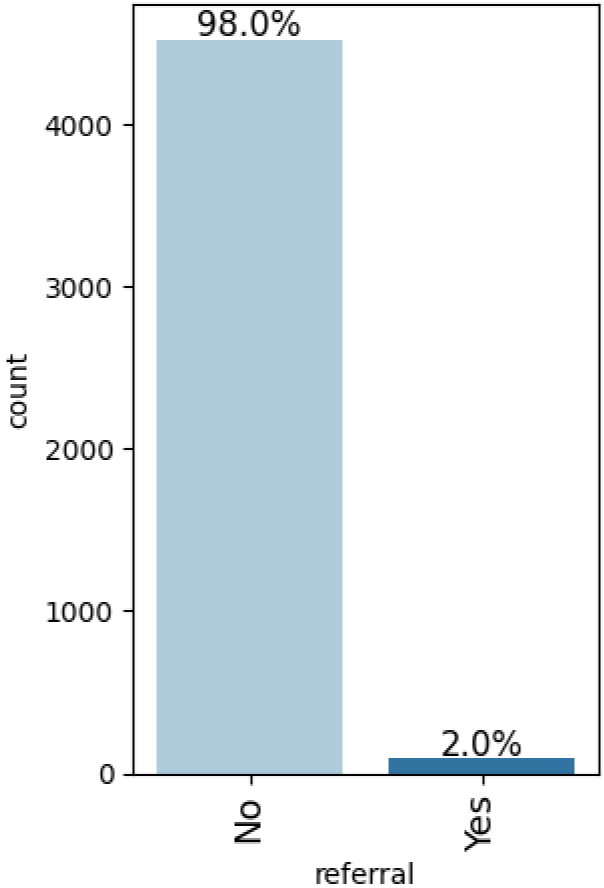

55.1%
44.9%
count
2500
2000
1500
1000
500
0
Mobile App
Website
first_interaction

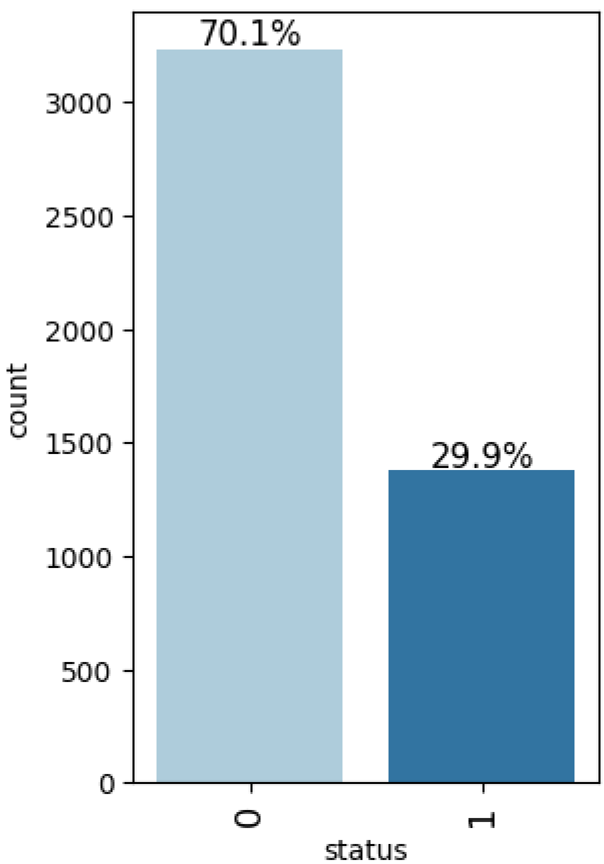

49.1%
48.6%
2.3%
count
2000
1500
1000
500
0
High
Low
Medium
profile_completed


89.2%
10.8%
count
4000
3500
3000
2500
2000
1500
1000
500
0
No
Yes
print_media_type1


88.6%
11.4%
count
No
Yes
digital_media


84.7%
15.3%
count
No
Yes
educational_channels





In [15]:
```
## BIVARIATE ANALYSIS
cols_list = edtech_leads.select_dtypes(include=np.number).columns.tolist()
```

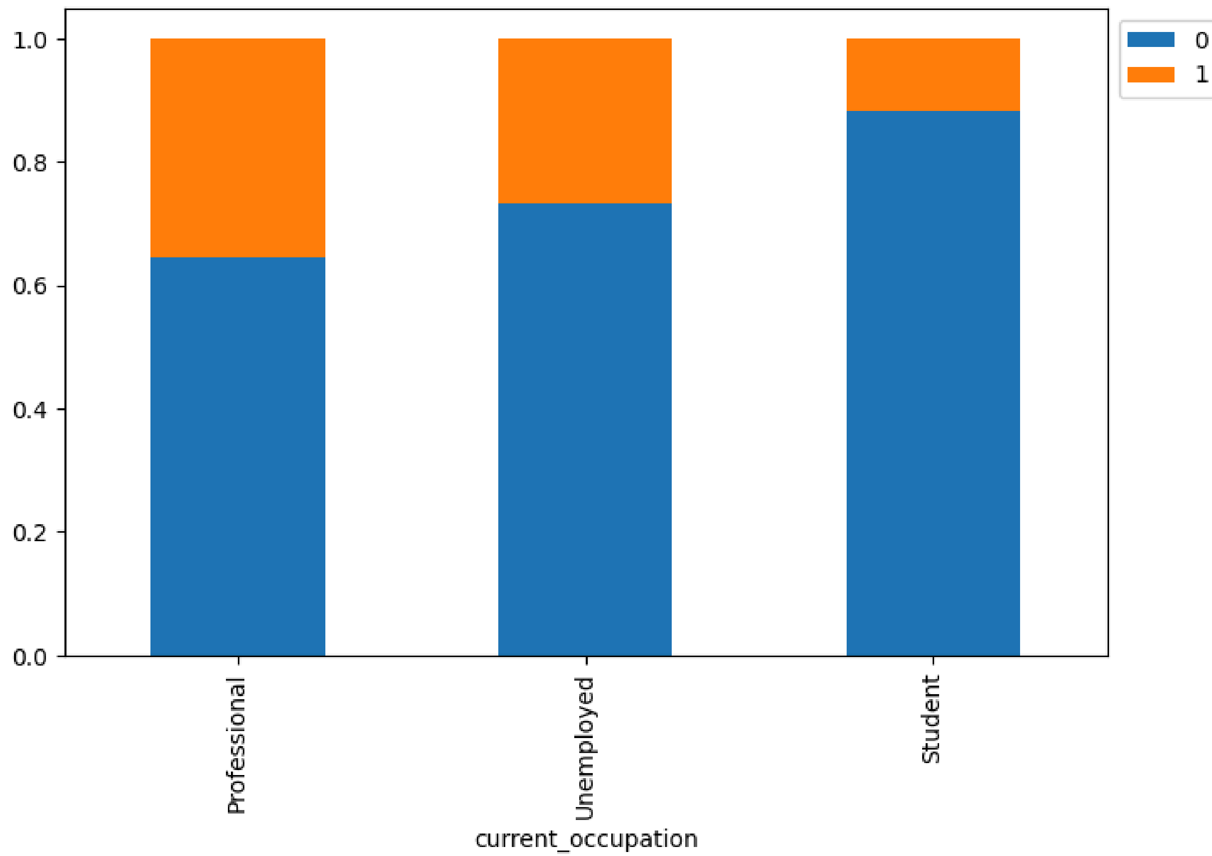
```
plt.figure(figsize=(10, 5))
sns.heatmap(
    edtech_leads[cols_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cma
)
plt.show()
```

In [16]:
```
## STACKED BARPLOT FOR BIVARIATE ANALYSIS
def stacked_barplot(edtech_leads, predictor, target):
    """
    Print the category counts and plot a stacked bar chart

    data: dataframe
    predictor: independent variable
    target: target variable
    """
    count = edtech_leads[predictor].nunique()
    sorter = edtech_leads[target].value_counts().index[-1]
    tab1 = pd.crosstab(edtech_leads[predictor], edtech_leads[target], margins=T
        by=sorter, ascending=False
    )
    print(tab1)
    print("-" * 120)
    tab = pd.crosstab(edtech_leads[predictor], edtech_leads[target], normalize=
        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()
```

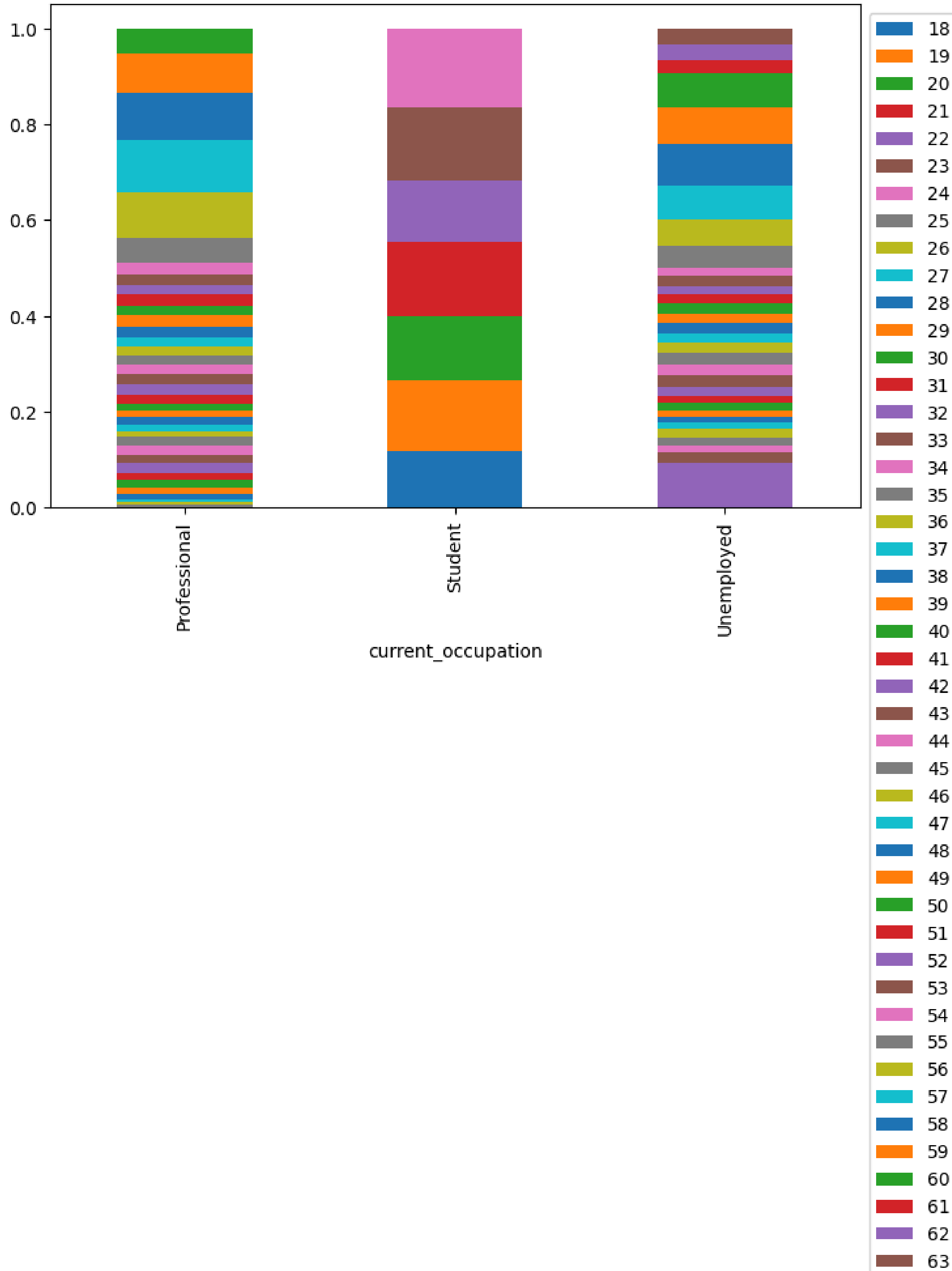
In [17]:
```
## BIVARIATE ANALYSIS THAT DESCRIBES LEAD DEMOGRAPHICS
stacked_barplot(edtech_leads, "current_occupation", "status") ## OCCUPATION V S
stacked_barplot(edtech_leads, "current_occupation", "age") ## OCCUPATION V AGE
```

```
stacked_barplot(edtech_leads, "age", "status") ## AGE V STATUS
## BIVARIATE ANALYSIS THAT SHOWS HOW LEADS RECEIVED ADVERTISEMENTS AND OR REFE
stacked_barplot(edtech_leads, "print_media_type1", "status")
stacked_barplot(edtech_leads, "print_media_type2", "status")
stacked_barplot(edtech_leads, "digital_media", "status")
stacked_barplot(edtech_leads, "referral", "status")
```

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

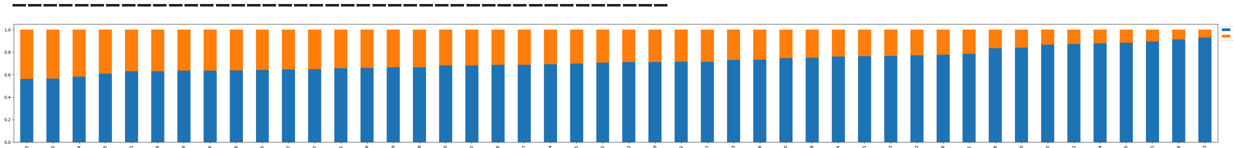


```
age                  18  19  20  21  22  23  24  25  26  27  28  29  30  31  \
current_occupation
Professional          0   0   0   0   0   0   0  16  15  14  27  36  44  38
All                  66  81  75  86  71  85  90  17  15  14  27  36  44  38
Student              66  81  75  86  71  85  90   1   0   0   0   0   0   0
Unemployed            0   0   0   0   0   0   0   0   0   0   0   0   0   0

age                   32  33  34  35  36  37  38  39  40  41  42  43  44  45  \
current_occupation
Professional          53  46  52  44  30  40  41  32  40  51  55  56  49  49
All                  188  76  74  66  58  60  58  52  63  70  83  89  81  84
Student                0   0   0   0   0   0   0   0   0   0   0   0   0   0
Unemployed           135  30  22  22  28  20  17  20  23  19  28  33  32  35

age                  46  47  48  49  50  51  52  53  54   55   56   57   58  \
current_occupation
Professional         53  52  56  61  53  60  53  59  65  134  252  282  256
All                  85  80  88  87  85  88  77  91  88  200  330  385  382
Student               0   0   0   0   0   0   0   0   0    0    0    0    0
Unemployed           32  28  32  26  32  28  24  32  23   66   78  103  126

age                   59   60  61  62  63   All
current_occupation
Professional         217  135   0   0   0  2616
All                  328  238  38  48  47  4612
Student                0    0   0   0   0   555
Unemployed           111  103  38  48  47  1441
-------------------------------------------------------------------------------
------------------------------------------
```

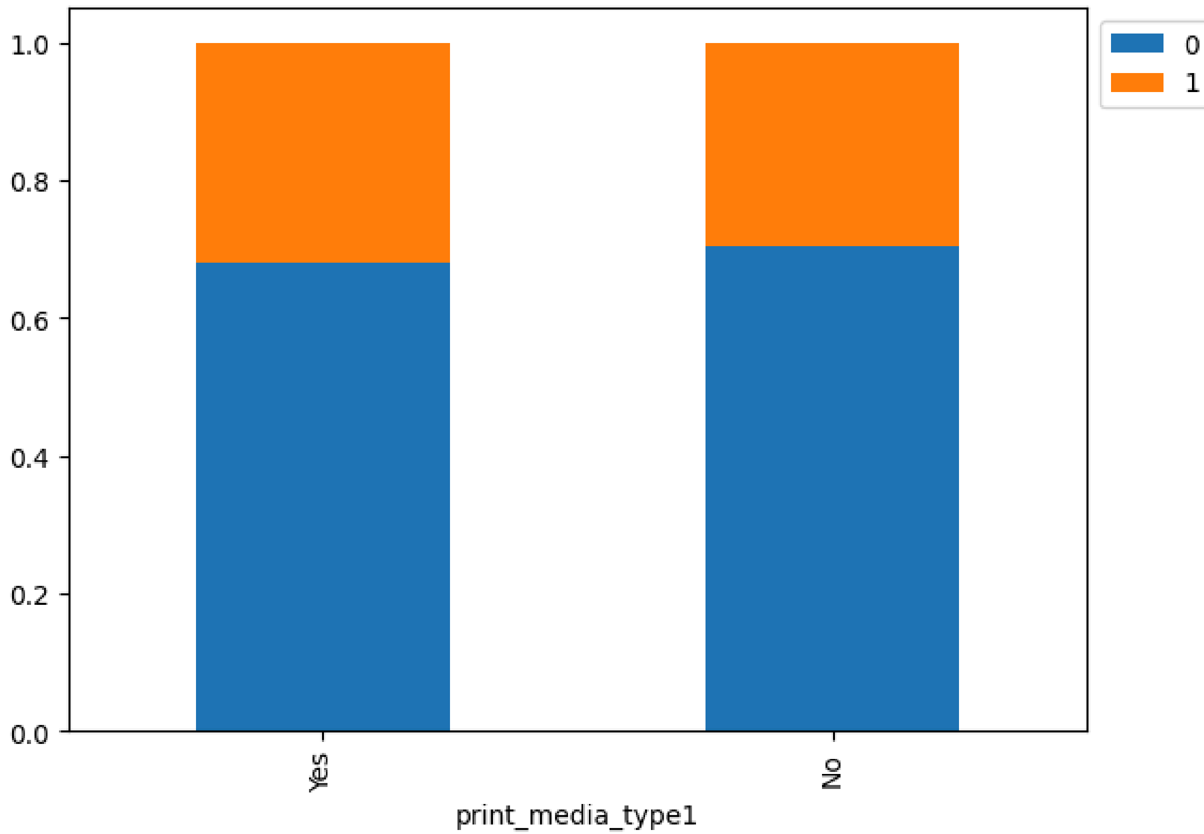

1.0
0.8
0.6
0.4
0.2
0.0
Professional
Student
Unemployed
current_occupation
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

status	0	1	All
age			
All	3235	1377	4612
57	249	136	385
59	207	121	328
58	262	120	382
56	210	120	330
60	162	76	238
55	128	72	200
32	133	55	188
53	51	40	91
50	48	37	85
45	51	33	84
34	43	31	74
46	56	29	85
41	44	26	70
47	55	25	80
44	56	25	81
49	62	25	87
42	59	24	83
43	65	24	89
48	66	22	88
52	55	22	77
36	37	21	58
37	39	21	60
54	67	21	88
35	46	20	66
39	33	19	52
51	69	19	88
40	47	16	63
30	30	14	44
31	25	13	38
38	45	13	58
29	24	12	36
20	63	12	75
24	79	11	90
63	36	11	47
62	37	11	48
18	55	11	66
33	66	10	76
28	18	9	27
22	62	9	71
21	77	9	86
61	29	9	38
19	74	7	81
23	79	6	85
26	11	4	15
27	10	4	14
25	15	2	17

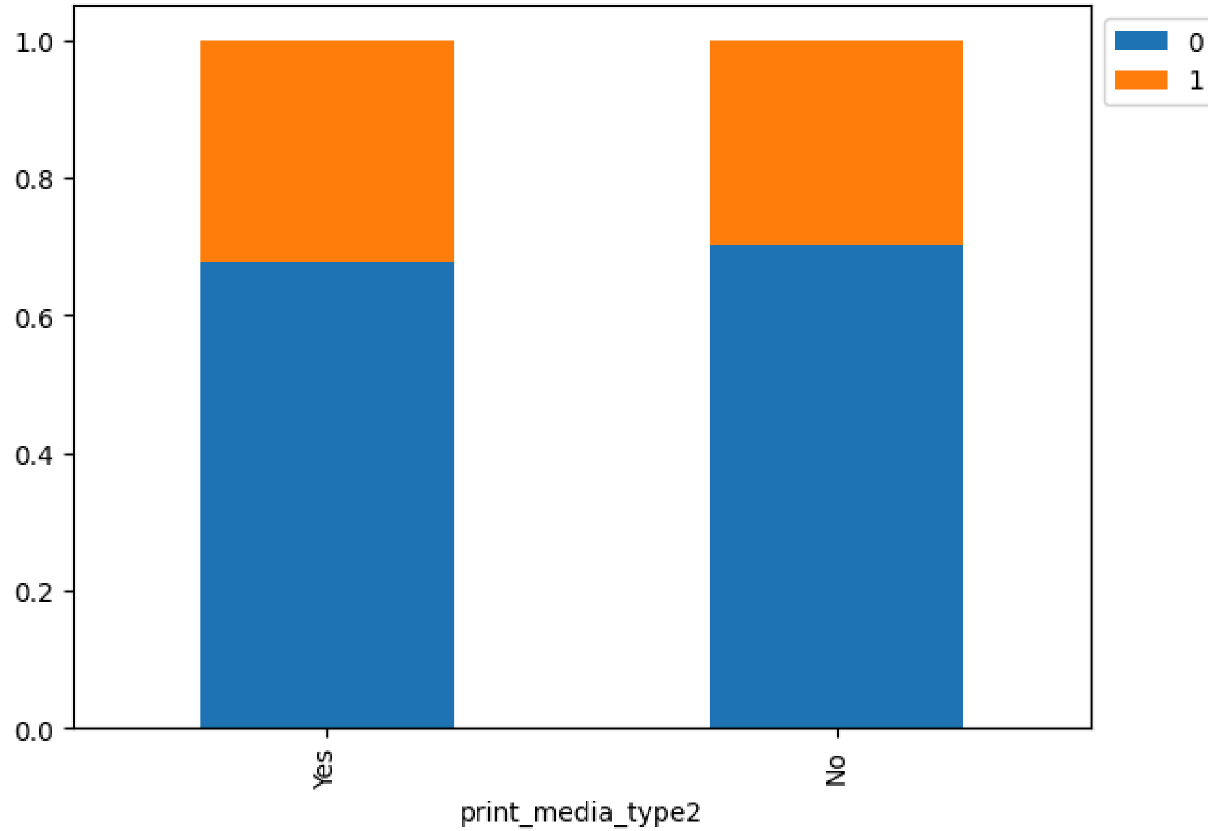
```
---------------------------------------------------------------------------------------------------------------------------
```



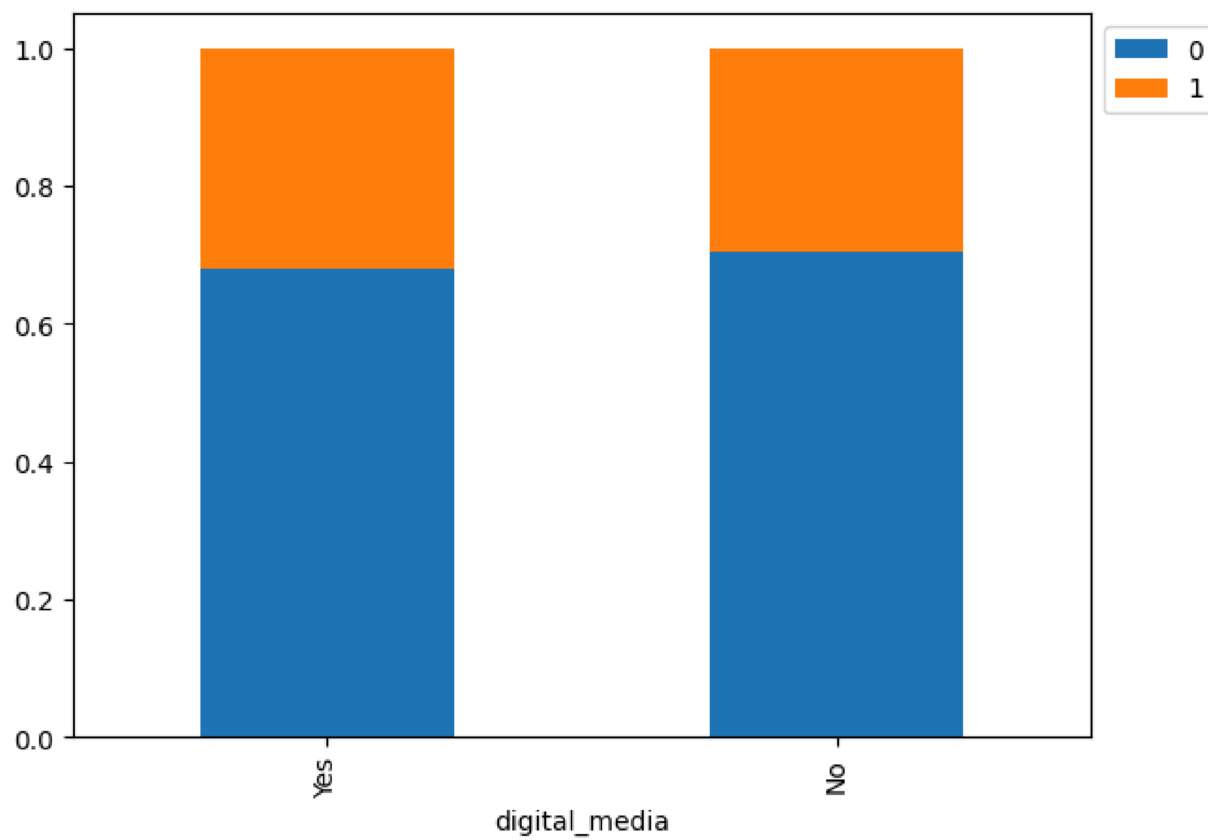
```
status                0     1   All
print_media_type1
All                3235  1377  4612
No                 2897  1218  4115
Yes                 338   159   497
------------------------------------------------------------------------------
------------------------------------------
```



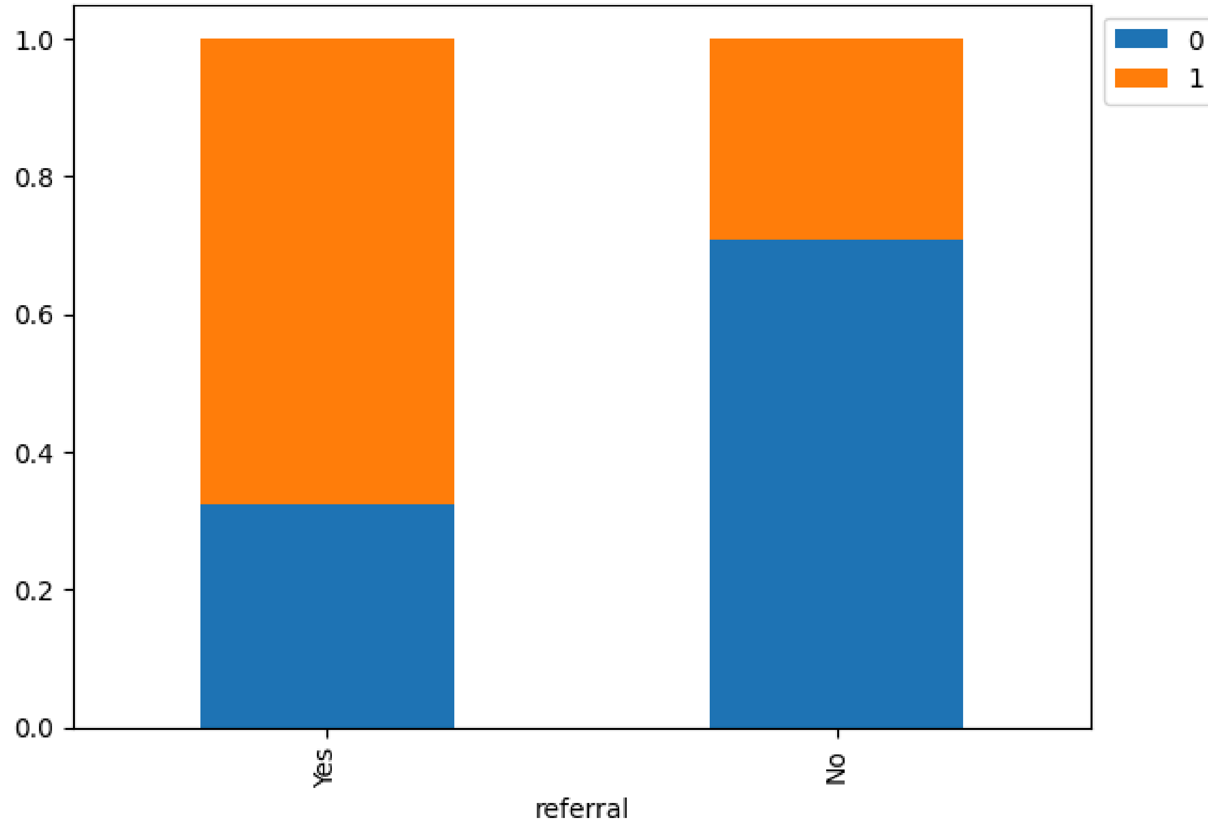

```
status                0     1   All
print_media_type2
All                3235  1377  4612
No                 3077  1302  4379
Yes                 158    75   233
------------------------------------------------------------------------------
------------------------------------------
```




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



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



Building a Decision Tree model

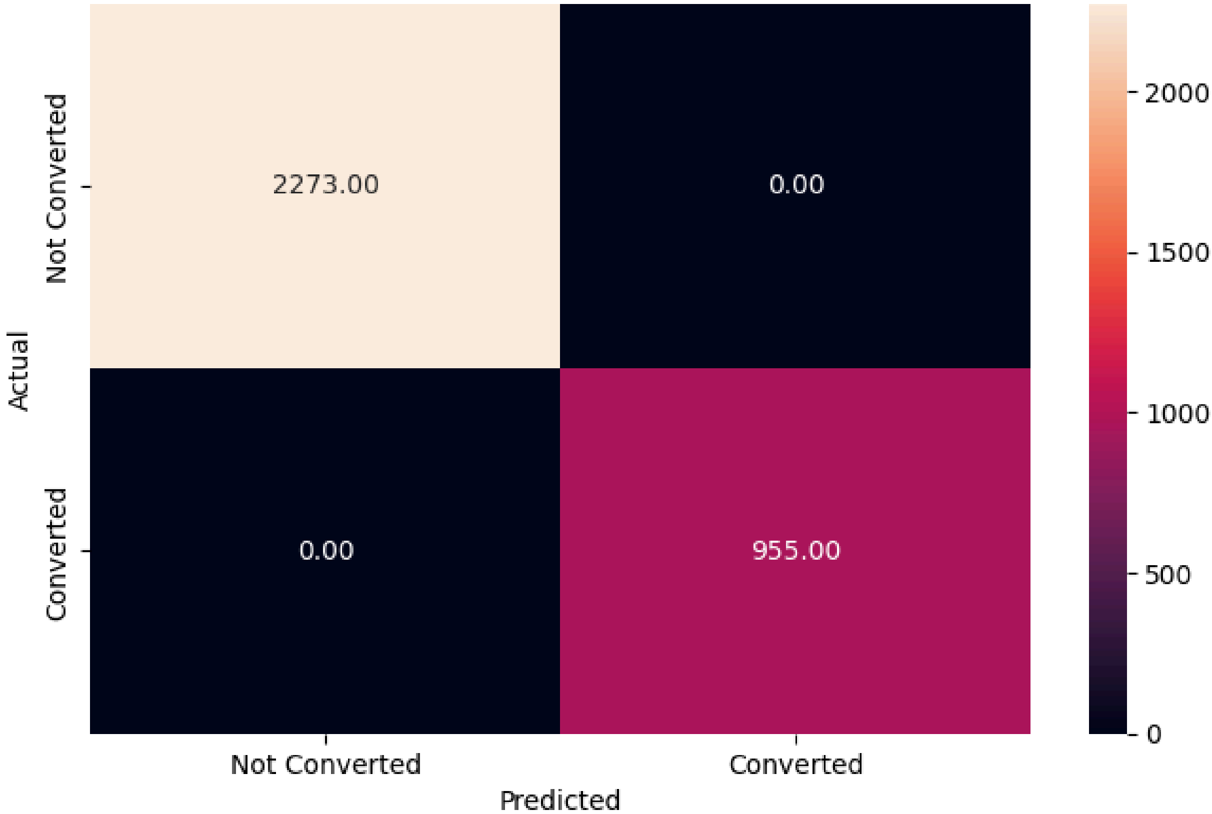
In [18]:
```
X = pd.get_dummies(X, drop_first=True)
Y = edtech_leads["status"] ## Coding dependent variable
# Split data 70:30 ]for train to test data
X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size=0.30, random_state=1
)
print("Shape of Training set : ", X_train.shape)
print("Shape 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, 4627)
Shape of test set :  (1384, 4627)
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
```

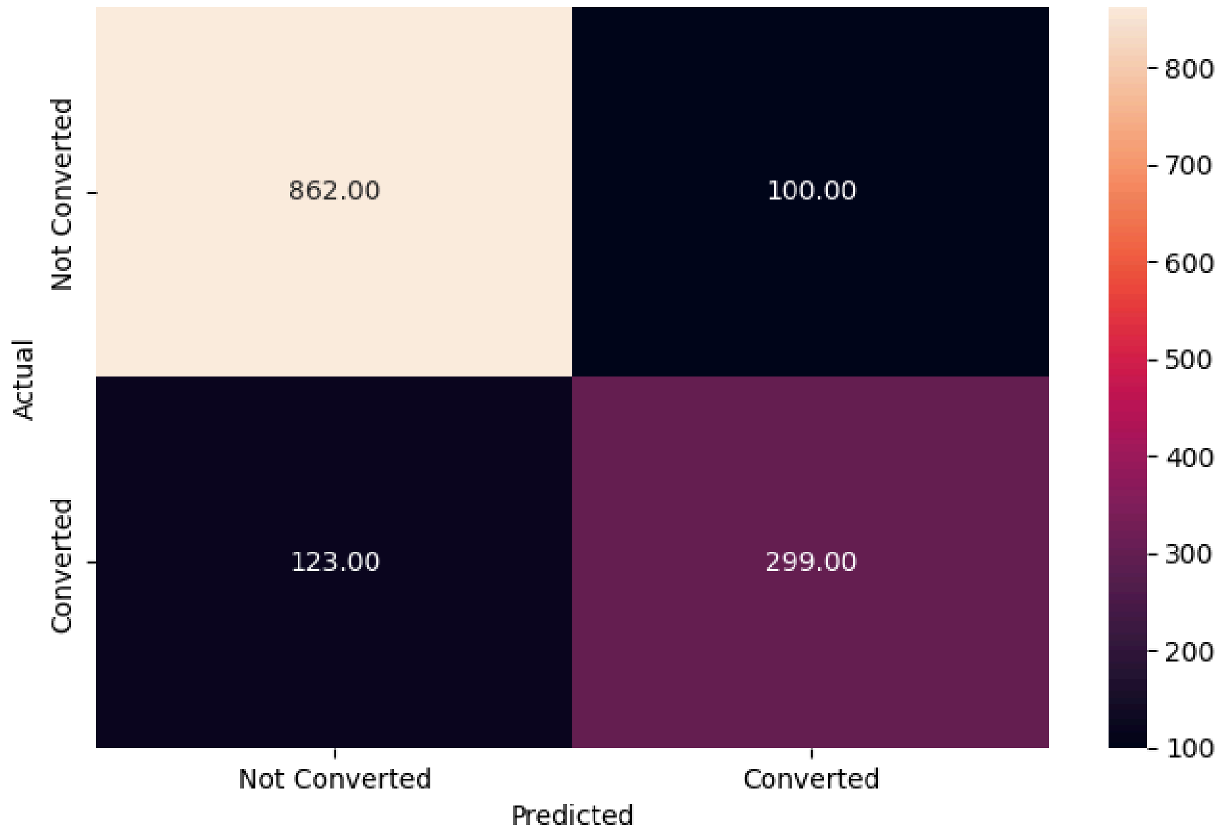
In [19]:
```
## CLASSIFICATION REPORT/BREAKDOWN

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'

    plt.ylabel('Actual')

    plt.xlabel('Predicted')

    plt.show()
```

In [20]:
```
## BUILDING DECISION TREE MODEL
d_tree = DecisionTreeClassifier()  # Create DecisionTreeClassifier
d_tree.fit(X_train, y_train)  # Fit classifier on training data
# performance check on training data
y_pred_train1 = d_tree.predict(X_train)
print("Performance on Training Data:")
metrics_score(y_train, y_pred_train1)
# performance check on testing data
y_pred_test1 = d_tree.predict(X_test)
print("Performance on Testing Data:")
metrics_score(y_test, y_pred_test1)
```

```
Performance on Training Data:
              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
```




```
Performance on Testing Data:
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       962
           1       0.75      0.71      0.73       422

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



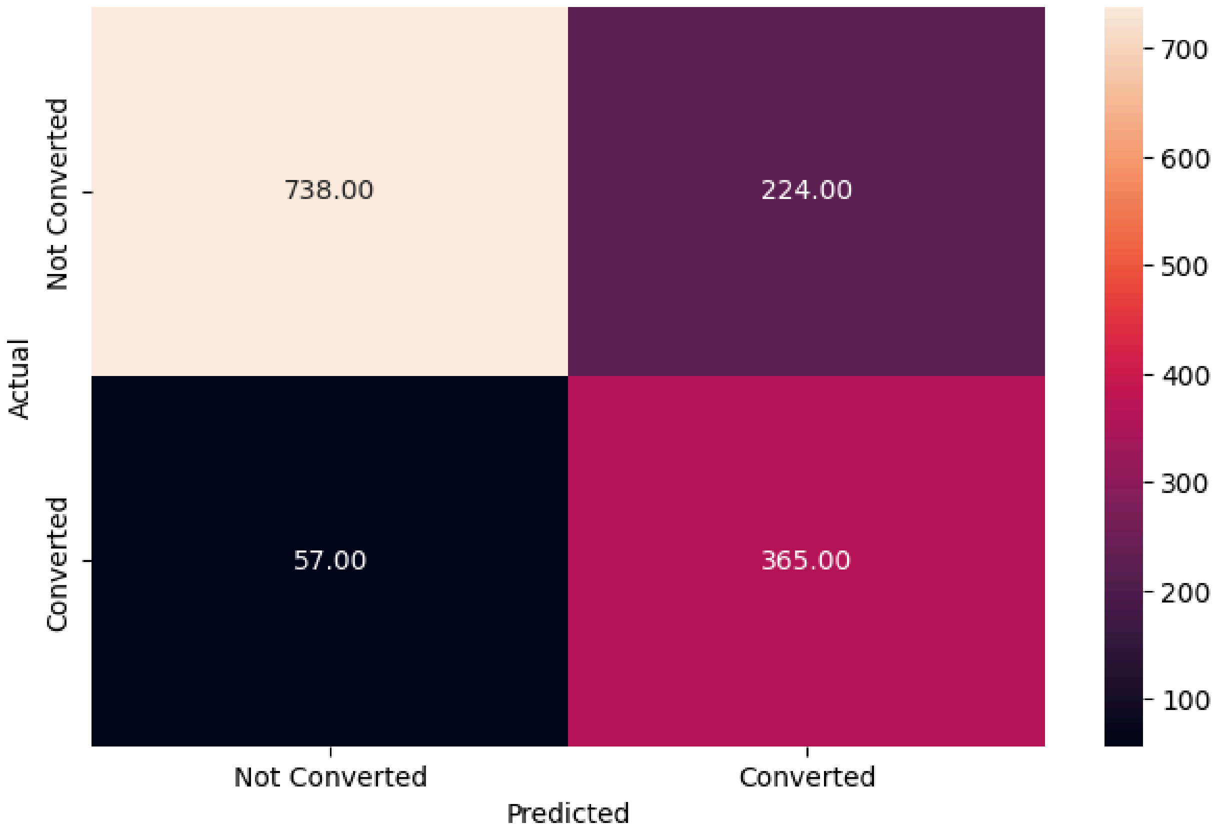
** OBSERVATIONS: The model performed extremely well on the training data, achieving perfect precision, recall, and accuracy. It is obvious that the model is overfitting the training data. This is also shown through the discrepancy between the performance on the training data & testing data.

In [21]:
```
# Choosing classifier type
d_tree_tuned = DecisionTreeClassifier(random_state = 7, class_weight = {0: 0.3
# parameter grid
parameters = {'max_depth': np.arange(2, 10),
              'criterion': ['gini', 'entropy'],
              'min_samples_leaf': [5, 10, 20, 25]
             }
#  scoring used to compare parameter combinations - recall score for class 1
scorer = metrics.make_scorer(recall_score, pos_label = 1)
# grid search
grid_obj = GridSearchCV(d_tree_tuned, parameters, scoring = scorer, cv = 5)
grid_obj = grid_obj.fit(X_train, y_train)
# 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)
```

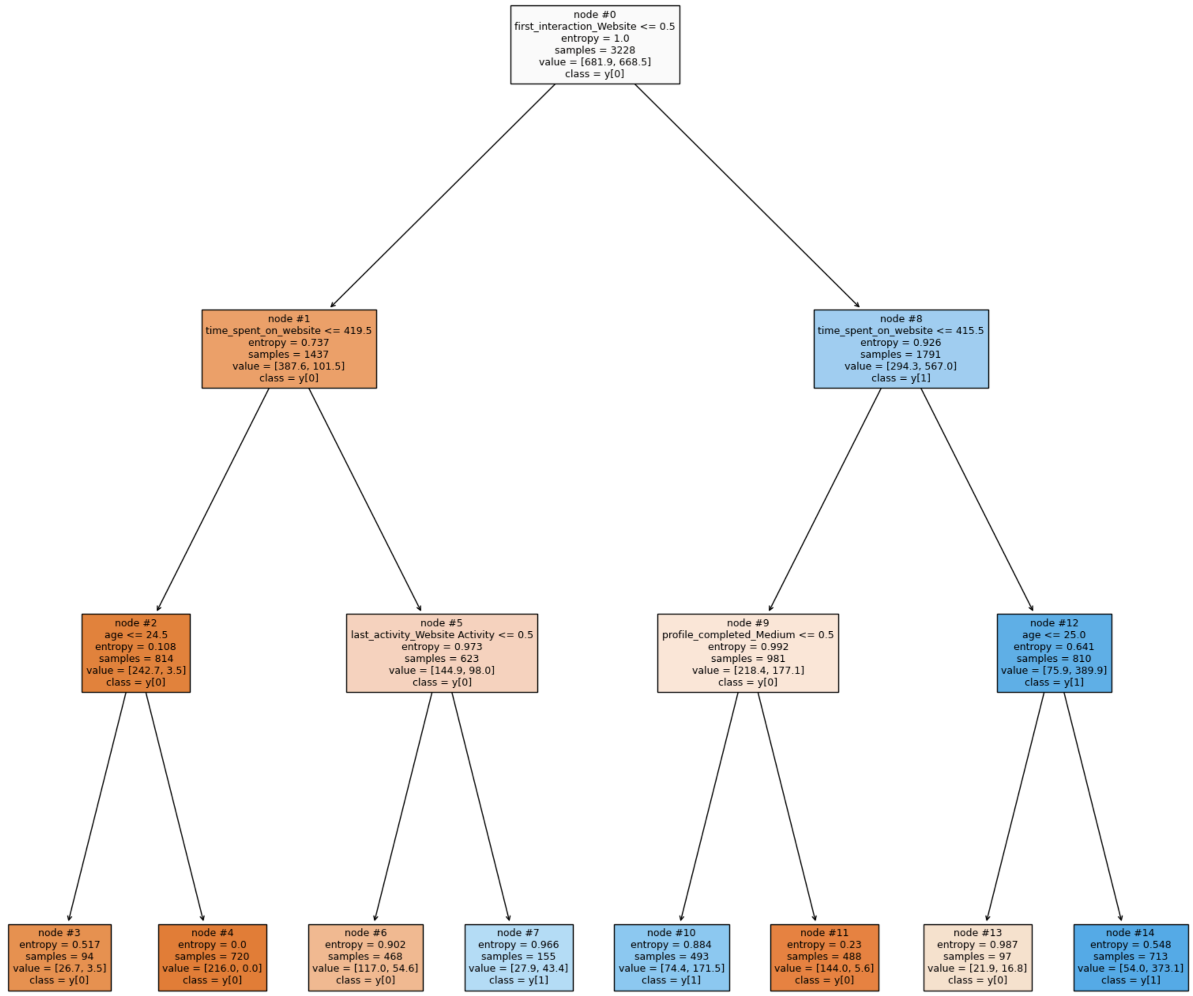
Out[21]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(class_weight={0: 0.3, 1: 0.7}, criterion='entro
py',
                       max_depth=3, min_samples_leaf=5, random_state=
7)
```

In [22]:
```
## Performance Check 2 on data
y_pred_train2 = d_tree_tuned.predict(X_train)
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
```



In [23]:
```
## VISUALISATION OF DECISION TREE
features = list(X.columns)

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

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

plt.show()
```



OBSERVATIONS: Most of the classes are split evenly. The original 3228 samples are split into two nodes, one containing apporximately 55% of the samples in node #8 and 45% of the samples in node #1. The criteria for the nodes/classes that the decision treep produced were based on entropy. The decision tree makes a lot of these decisions with a high calculated entropy. 3 out of the final 8 nodes had an entropy greater than 0.9. While node #11 had an entropy of 0.23, node #4 had an entropy of 0.0, indicating full purtiy amongst that node.

Do we need to prune the tree?

Yes the tree should be pruned since many of the classes were made with high uncertainty.

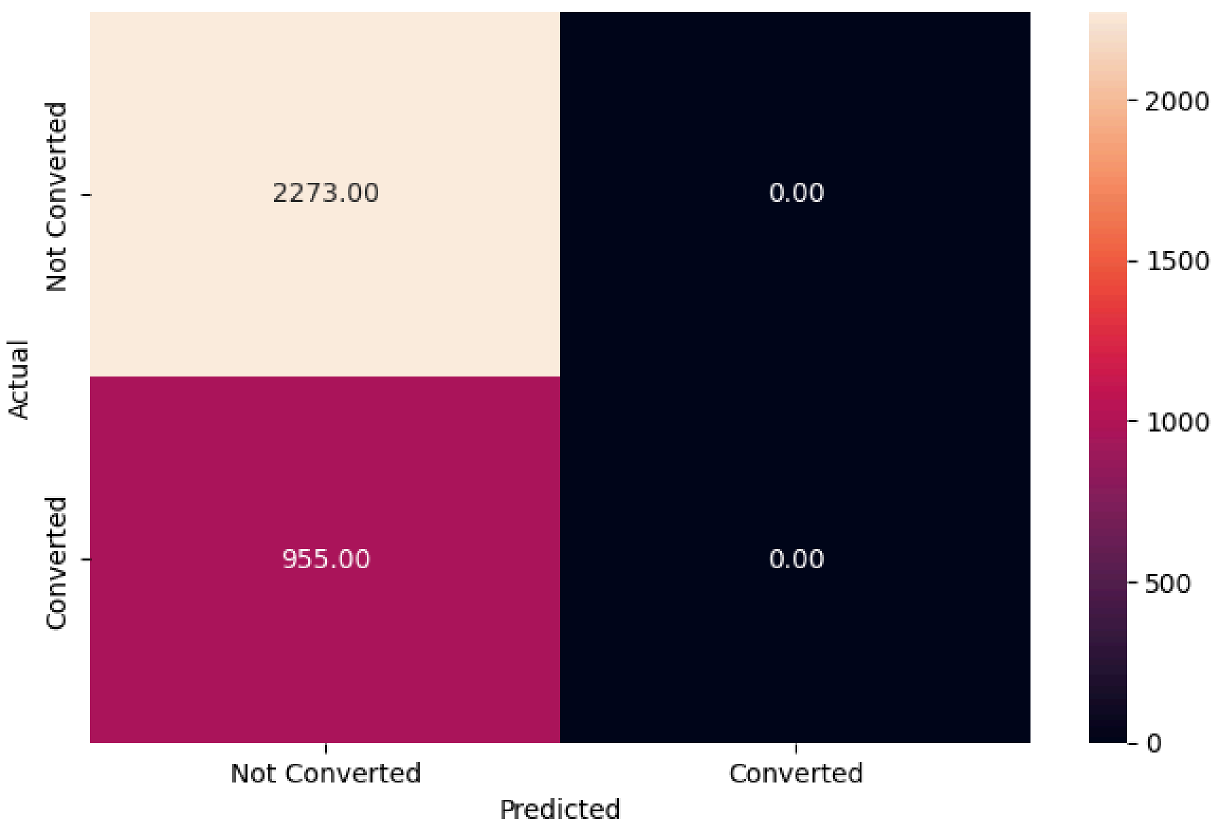
Building a Random Forest model

In [24]:
```
# Fitting the random forest tree classifier on the training data
rf_estimator = RandomForestClassifier(n_estimators=100, max_depth=5, min_sampl
rf_estimator.fit(X_train, y_train)
```

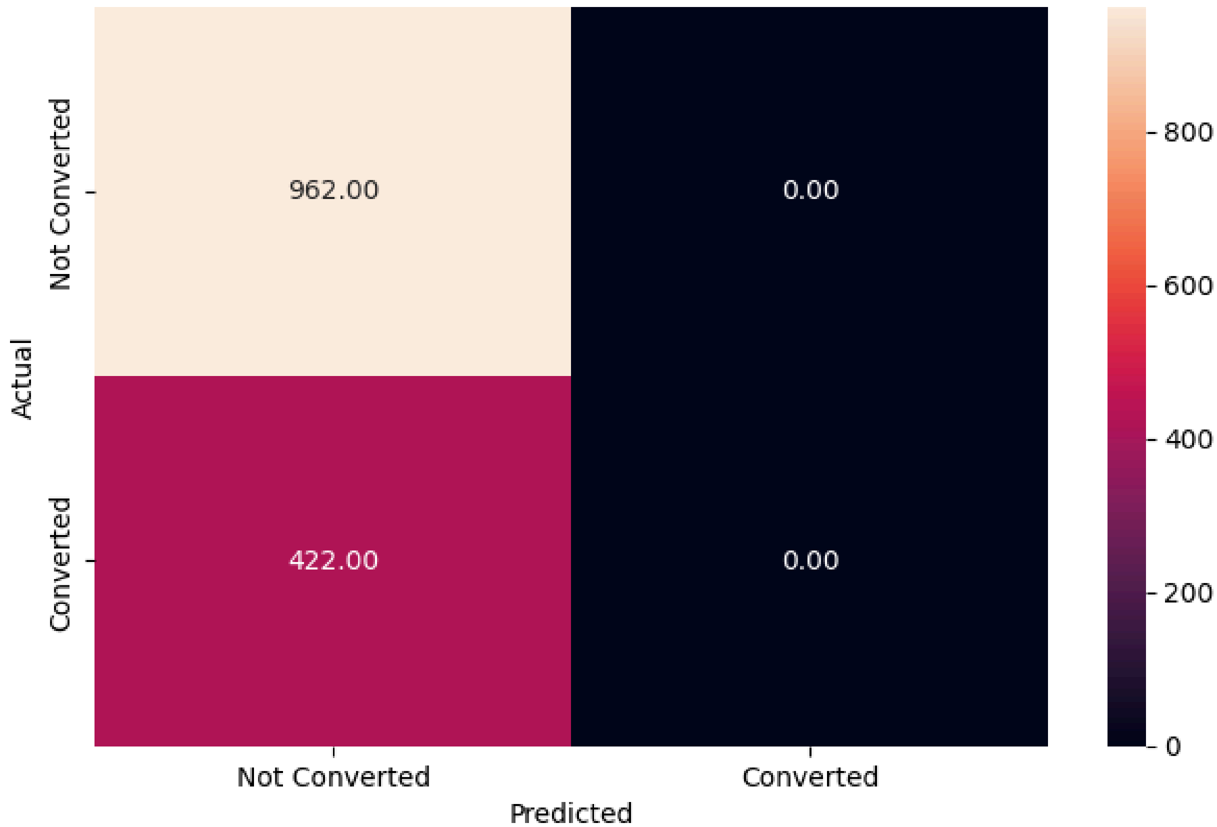
Out[24]:
```
RandomForestClassifier
RandomForestClassifier(max_depth=5)
```

In [25]:
```
# predictions on the training data
y_pred_train3 = rf_estimator.predict(X_train)
# Checking performance on the testing data
y_pred_test3 = rf_estimator.predict(X_test)

print("Performance on Training Data:")
metrics_score(y_train, y_pred_train3)

print("Performance on Testing Data:")
metrics_score(y_test, y_pred_test3)
```

```
Performance on Training Data:
              precision    recall  f1-score   support

           0       0.70      1.00      0.83      2273
           1       0.00      0.00      0.00       955

    accuracy                           0.70      3228
   macro avg       0.35      0.50      0.41      3228
weighted avg       0.50      0.70      0.58      3228
```



```
Performance on Testing Data:
              precision    recall  f1-score   support

           0       0.70      1.00      0.82       962
           1       0.00      0.00      0.00       422

    accuracy                           0.70      1384
   macro avg       0.35      0.50      0.41      1384
weighted avg       0.48      0.70      0.57      1384
```



** OBSERVATIONS: Zeros in class one suggest extremely poor performance, and the need for more fine tuning and even pruning, since the model is highly uniterpretable.

Actionable Insights and Recommendations

It would be recommended that lots of fine tuning and pruning would needed to be done on these models for greater insight. There are high levels of entropy, and more importantly low recall, precision, and F-1 scores for many of the classes, in many of the tests there were values of zero registered for those categories.

In terms of values that stood out & held significance;

- Referrals were the lowest source of leads — only 2%
- Time spent on website had the highest correlation to lead status out of all numerical categories
- The distribution of the age of leads is strongly skewed to the left

Age being highly skewed to the left is something that should be explored more. It was extremely glaring how the majority of leads were in their late 50s. This could suggest a variety of potential explanations, the most logical one would be the fact that people in their late 50s might be seeking education opportunities for their children, who are not finanically capable of paying for tuition themselves. It could also suggest people in their late 50s wanting to upskill. It is important to find out why this is since the age demographic is clearly not uniform and an important facet to establishing reliable leads.

In [34]:
```
!pip install nbconvert
!jupyter nbconvert --to html Optimizing_EdTech_Lead_Conversion_Rates_SpencerRog
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-pac
kages (6.5.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dis
t-packages (from nbconvert) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packag
es (from nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-pa
ckages (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.1
0/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-p
ackages (from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.10/
dist-packages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.1
0/dist-packages (from nbconvert) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/di
st-packages (from nbconvert) (2.1.5)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.10/
dist-packages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.10/di
st-packages (from nbconvert) (0.10.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist
-packages (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-pac
kages (from nbconvert) (24.0)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.
10/dist-packages (from nbconvert) (1.5.1)
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.10/di
st-packages (from nbconvert) (2.16.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-pack
ages (from nbconvert) (1.3.0)
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.10/dis
t-packages (from nbconvert) (5.7.1)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/
dist-packages (from jupyter-core>=4.7->nbconvert) (4.2.1)
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python
3.10/dist-packages (from nbclient>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.
10/dist-packages (from nbformat>=5.1->nbconvert) (2.19.1)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.10/di
st-packages (from nbformat>=5.1->nbconvert) (4.19.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist
-packages (from beautifulsoup4->nbconvert) (2.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-pa
ckages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist
-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/lo
cal/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconve
rt) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.1
0/dist-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dis
t-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.18.1)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.10/dist-pac
```

```
kages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (24.0.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.
10/dist-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.
8.2)
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.10/dist-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.3.3)
[NbConvertApp] Converting notebook Optimizing_EdTech_Lead_Conversion_Rates_Spe
ncerRogovin_.ipynb to html
[NbConvertApp] Writing 1611051 bytes to Optimizing_EdTech_Lead_Conversion_Rate
s_SpencerRogovin_.html
```