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.

Importing necessary libraries

In [181…
```
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 different 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,
)

from sklearn.utils.class_weight import compute_class_weight
from sklearn.model_selection import learning_curve
```

Import Dataset

In [182…
```
learn = pd.read_csv("ExtraaLearn.csv")
```

In [183…
```
# 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 [184…
```
data.head()
```

Out[184...

	ID	age	current_occupation	first_interaction	profile_completed	website_visits	time_spent_on_website	page_views_per_vis
0	EXT001	57	Unemployed	Website	High	7	1639	1.861(
1	EXT002	56	Professional	Mobile App	Medium	2	83	0.320(
2	EXT003	52	Professional	Website	Medium	3	330	0.074(
3	EXT004	53	Unemployed	Website	High	4	464	2.057(
4	EXT005	23	Student	Website	High	4	600	16.914(

In [185...

```
data.tail()
```

Out[185...

	ID	age	current_occupation	first_interaction	profile_completed	website_visits	time_spent_on_website	page_views_pe
4607	EXT4608	35	Unemployed	Mobile App	Medium	15	360	2
4608	EXT4609	55	Professional	Mobile App	Medium	8	2327	5
4609	EXT4610	58	Professional	Website	High	2	212	2
4610	EXT4611	57	Professional	Mobile App	Medium	1	154	3
4611	EXT4612	55	Professional	Website	Medium	4	2290	2

Understand the shape of the dataset

In [186...

```
data.shape
```

Out[186...

```
(4612, 15)
```

In [187...

```
# checking for duplicate values
data["ID"].nunique()
```

Out[187...

```
4612
```

In [188...

```
# checking for duplicate values
data.duplicated().sum()
```

Out[188...

```
np.int64(0)
```

Observations:

- The dataset has no duplicated record.

Check the data types of the columns for the dataset

In [189...

```
data.dtypes
```

Out[189...

```
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
```

Exploratory Data Analysis

Statistical summary of the data.

In [190…
```
data.describe().T
```

Out[190…

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

List of count of each unique value in each column

In [191…
```
# Making a list of all catrgorical variables except ID
cat_col = list(data.drop(["ID"], axis = 1).select_dtypes("object").columns)

# Printing number of count of each unique value in each column
for column in cat_col:
    print(data[column].value_counts())
    print("-" * 50)
```

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

Univariate Analysis on Numerical Features

Boxplot will be used to visualize the distribution and histogram displays the frequency distribution of the dataset.

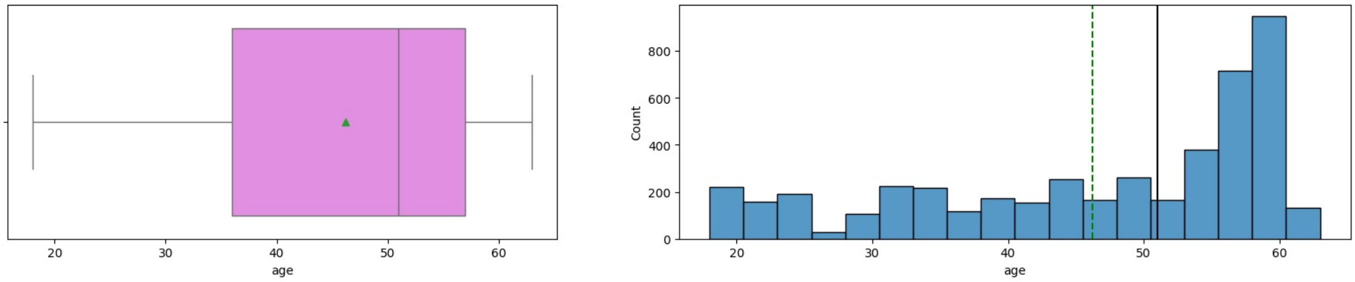
In [192…
```
#Remark: the following codes are reutilized from Low Code Version - Potential Customers Prediction.
# Modication: Show boxplot and histogram alongside in a row
# function to plot a boxplot and a histogram along the same scale.
def histogram_boxplot(data, feature, figsize=(20, 3.5), kde=False, bins=None):
    """
```

```
    Boxplot and histogram combined
    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (20,3.5))
    kde: whether to the show density curve (default False)
    bins: number of bins for histogram (default None)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        ncols=2,  # Number of columns of the subplot grid = 2
        gridspec_kw={
            "width_ratios": (0.45, 0.55),  # increased boxplot width ratio
            "wspace": 0.2  # reduced spacing between plots
        },
        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 column

    sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2, bins=bins, palette="winter"
    ) 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
```

Distribution on age

In [193…

```
histogram_boxplot(data, "age")
```



Observations:

Boxplot:

- 50% leads (IQR) have a age range of roughly between 35-58.
- No significant outliers are found.

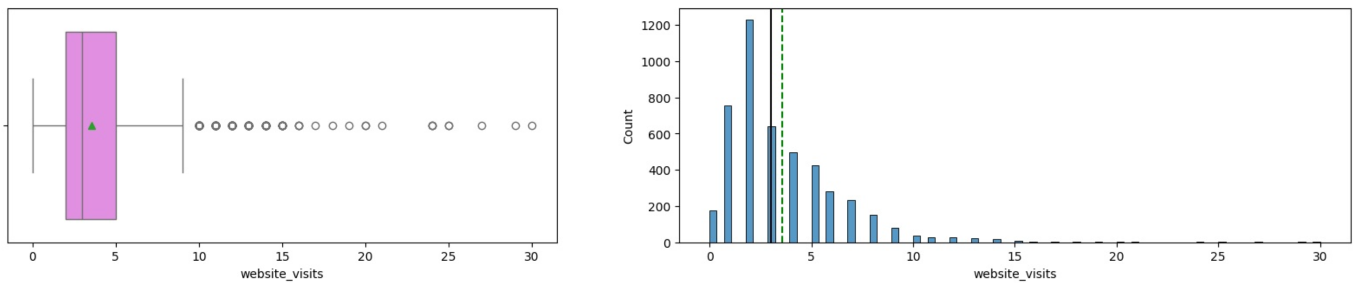
Histyogram:

- The highest concentration of ages range is 50-60.
- There's a lower density of ages in the range 25-30.
- There's a moderate concentration in the range 30-50.

Distribution on website_visits

In [194…

```
histogram_boxplot(data, "website_visits")
```



Observations:

Boxplot:

- The IQR is narrow, approximately between 2-5 visits.
- There are many outliers extending from 10 to 30 visits.

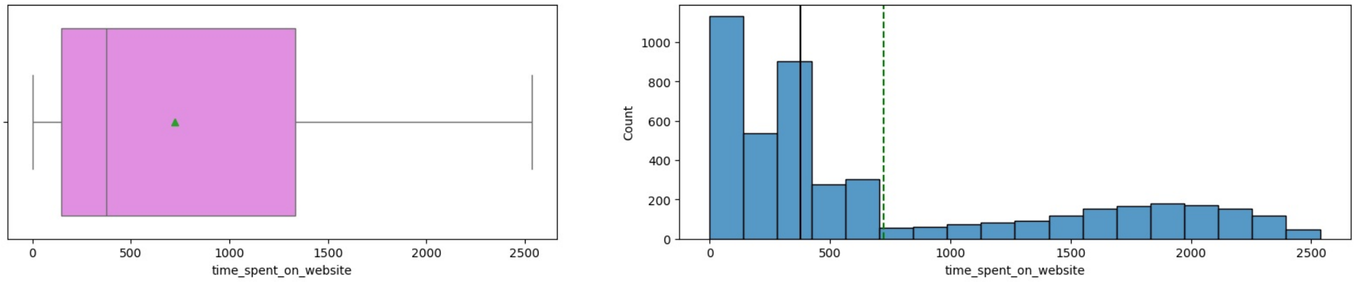
Histyogram:

- The majority of leads have 0-5 website visits, with a mean aropund 4.
- Website vists drop significantly after 5 times.

Distribution on time_spent_on_website

In [195…

```
histogram_boxplot(data, "time_spent_on_website")
```



Observations:

Boxplot:

- 50% of leads (IQR) have time spent on website in the range 300 - 1400, with min 0 and max 2,500.
- No outliers are found.

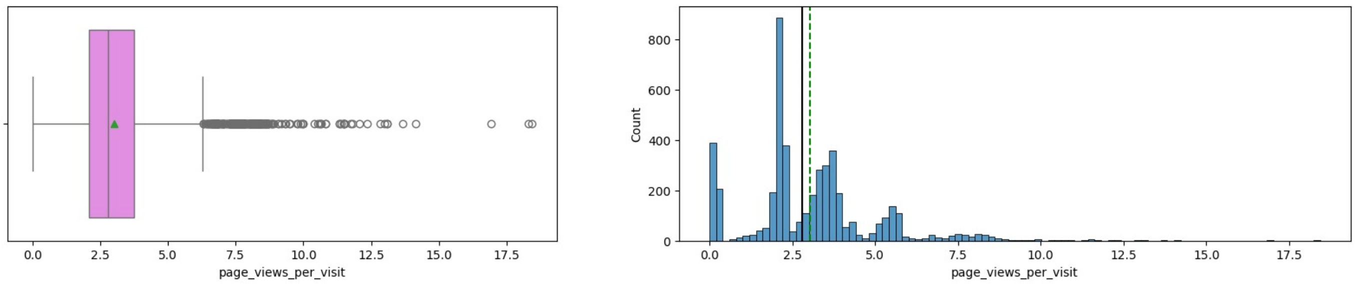
Histogram:

- The distribution is right-skewed, with most users spending time in the range 0 - 600, with some users spending more time (1500-2500).
- High concentration of leads have time spent on website in the lower time ranges (0-500 minutes).

Distribution on page_views_per_visit

In [196…

```
histogram_boxplot(data, "page_views_per_visit")
```



Observations:

Boxplot:

- The majority of users (50%) view have 2-5 pages per visit.
- The most outliers contracted between 5.5 to 12.5.
- There are some outliers with very high page views over 10, suggesting highly engaged users or potential technical issues.

Histogram:

- The distribution is right-skewed, with most users having between 1.5-5.5 page views per visit.
- Page views drop sharply after about 6 pages view

Univariate Analysis on Categorical Features

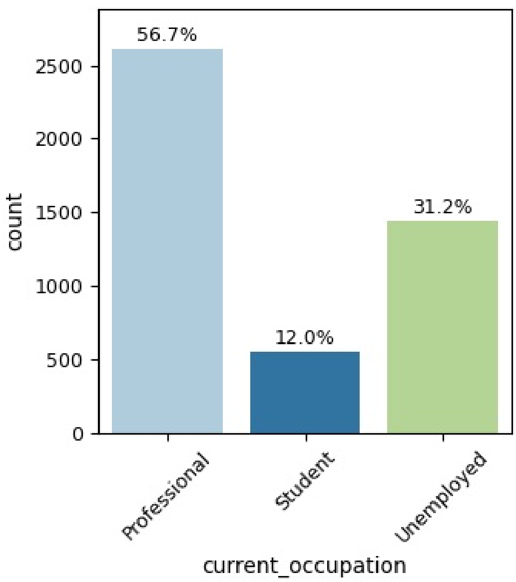
In [197…

```
#Remark: the following codes are reutilized from Low Code Version - Potential Customers Prediction.
# function to create labeled barplots
def labeled_barplot(data, feature, perc=False, n=None):
    """
    Barplot with percentage at the top

    Parameters:
    -----------
    data: dataframe
        Input dataframe
    feature: str
        Dataframe column to plot
    perc: bool, default False
```

```
        Whether to display percentages instead of count
    n: int, optional
        Displays the top n category levels (default is None, i.e., display all levels)
    """
    total = len(data[feature])  # length of the column
    count = data[feature].nunique()
    if n is None:
        # Reduce figure size by 40%
        plt.figure(figsize=((count + 1) * 0.9, 4))
    else:
        plt.figure(figsize=((n + 1) * 0.9, 4))

    # Adjust font sizes
    plt.xticks(rotation=45, fontsize=9)
    plt.yticks(fontsize=9)

    ax = sns.countplot(
        data=data,
        x=feature,
        palette="Paired",
        order=data[feature].value_counts().index[:n].sort_values(),
    )

    # Adjust title and label font sizes
    ax.set_xlabel(feature, fontsize=10)
    ax.set_ylabel("count", fontsize=10)

    # Set new y-axis limits with buffer space at the top
    y_min, y_max = ax.get_ylim()
    buffer = 0.05 * y_max
    ax.set_ylim(y_min, y_max + buffer)

    for p in ax.patches:
        if perc:
            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_width() / 2 + p.get_x()  # width of the plot
        y = p.get_height()  # height of the plot

        ax.annotate(
            label,
            (x, y),
            ha="center",
            va="center",
            size=9,  # Reduced annotation font size
            xytext=(0, 6),  # Reduced offset
            textcoords="offset points"
        )  # annotate the percentage

    plt.tight_layout()  # Ensure plot fits in the figure
    plt.show()  # show the plot
```

Distribution on current_occupation

In [198…
```
labeled_barplot(data, "current_occupation", perc=True)
```



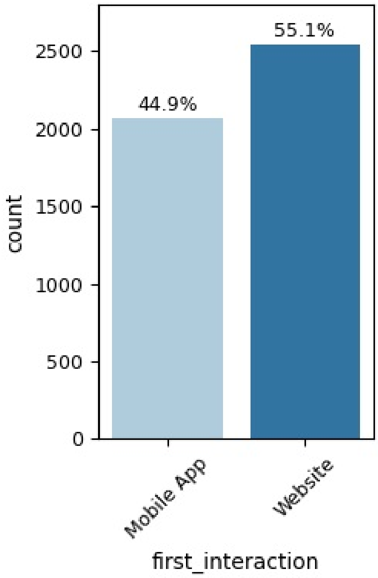
Observations:

- Over half (56.7%) of leads are professional, followed by unemployed (31.2%) and only small portion (12%) of them are students.
- Professional leads are more afforable to pay for online education.
- Unemployed leads are more encouraged to have online education because of better job market opportunities.
- Low percentages of students because students generally are not affordable for online education.

Distribution on first_interaction

In [199…

```
labeled_barplot(data, "first_interaction", perc=True)
```



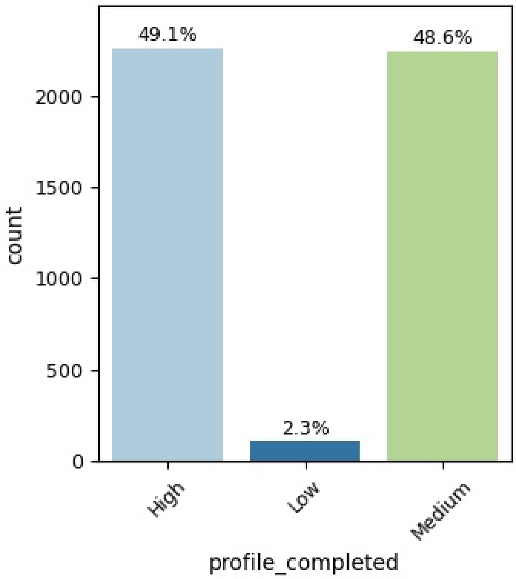
Observations:

- Both mobile app and website are important channels for leads engagements.
- Website is the primary first interaction point and mobile app is also strong.
- This means the they should be maintained for consistent user experiences because they're main entry points for users.

Distribution on profile_completed

In [200…

```
labeled_barplot(data, "profile_completed", perc=True)
```

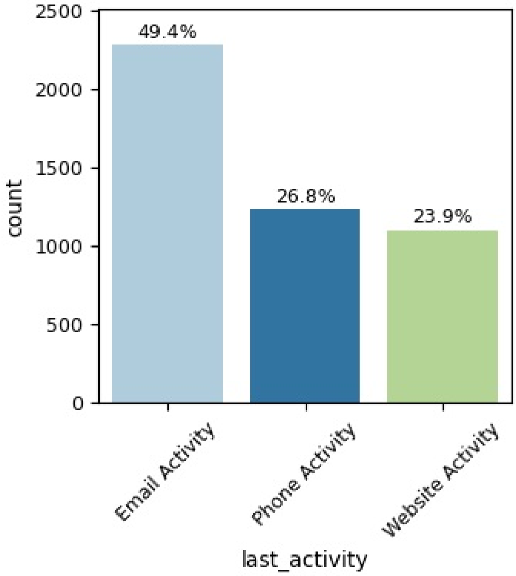


Observations:

- High and medium completetion levels are almost equally high, and very proportion of leads have low profile completion. The profile completion rate is overall very well.
- The overall profile completion rate is well enough and only very fewleadss leave their profiles in ExtraaLearn empty.
- High profile completion rates mean opportunities to utilize the profile data for personalized user experiences.
- There is a need to investigate into any barries fleadsers of low profile completion rates.
- More should be done to convert medium completion profiles ito ..high

Observations on last_activity

In [201…
```
labeled_barplot(data, "last_activity", perc=True)
```

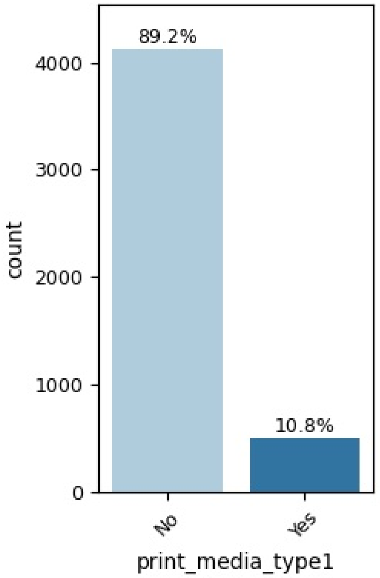


Observations:

- Email activiy is most important final touch point in leads engagement, followed by phone and website.
- High percentage of email activity means the email marketing is effective.
- Substantial phone activity indicates importance of personal contact.
- Also, substantial website activity means importance of online information for users.
- As the website activty is lowest, website engagement should be improved because this could encourage real courses enrollment.

Distribution on print_media_type1 (Newspaper Advertisements)

In [202…
```
labeled_barplot(data, "print_media_type1", perc=True)
```

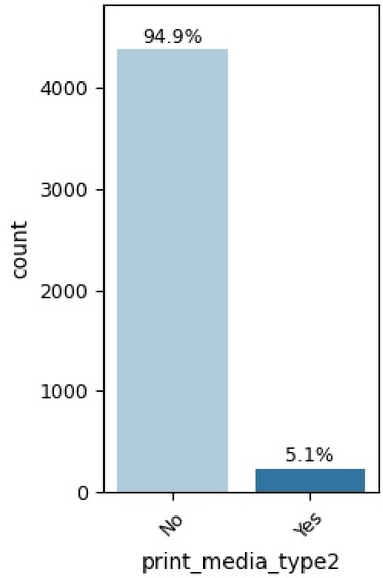


Observations:

- Only 10.8% leads reached through this media channel.
- Obviously this channel is not effective for leads engagement.
- As importance of paper media is declining in modern world, the management should review the resources allocation pent on newspaper advertistments for better conversion effectiveness.

Distribution on print_media_type2 (Magazine Advertisements)

In [203…
```
labeled_barplot(data, "print_media_type2", perc=True)
```

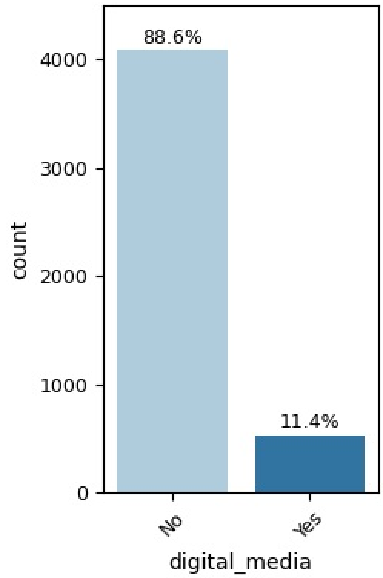


Observations:

- Similar to newspapers.

Distribution on digital_media

In [204…
```
labeled_barplot(data, "digital_media", perc=True)
```

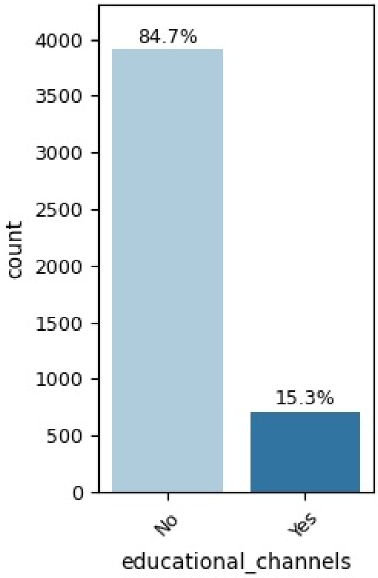


Observations:

- The pattern is almost the same as newspapers and magazines.
- Assume digital media means social media such as facebook, instragram, twitter, youtube or online advertisments, and lead engagements mean that leads access ExtraaLearn information. The above plot shows that leads engagement with digital media is very low.
- Unlike newspaper and magzine channels, the effectiveness of digital media should be reviewed and improved because digital media is a main trend in modern marketing.

Distribution on educational_channels

In [205…
```
labeled_barplot(data, "educational_channels", perc=True)
```



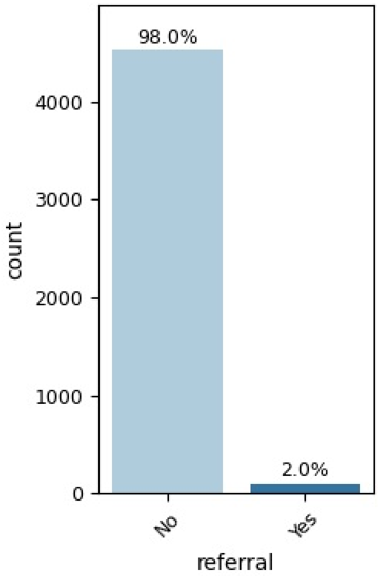
Observations:

- The pattern is almost the same as digital media.
- The effectiveness of educational channels should be reviewed and improved.

Distribution on referral

In [206…

```
labeled_barplot(data, "referral", perc=True)
```



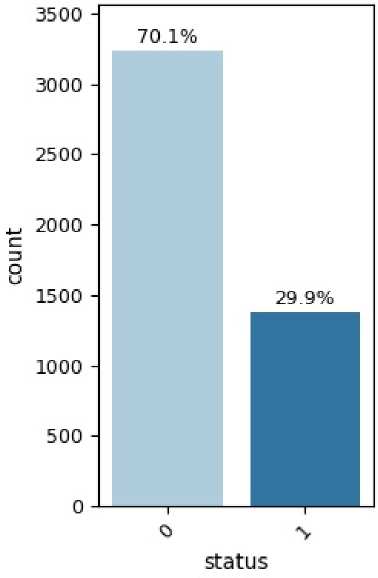
Observations:

- Only 2% leads heard about ExtraaLearn means its popularity should be improved.
- More resources (e.g, referral fee or discount) should be used to promote referral as words of mouth is a effective way to attract potential customers.

Distribution on status

In [207…

```
labeled_barplot(data, "status", perc=True)
```



Observations:

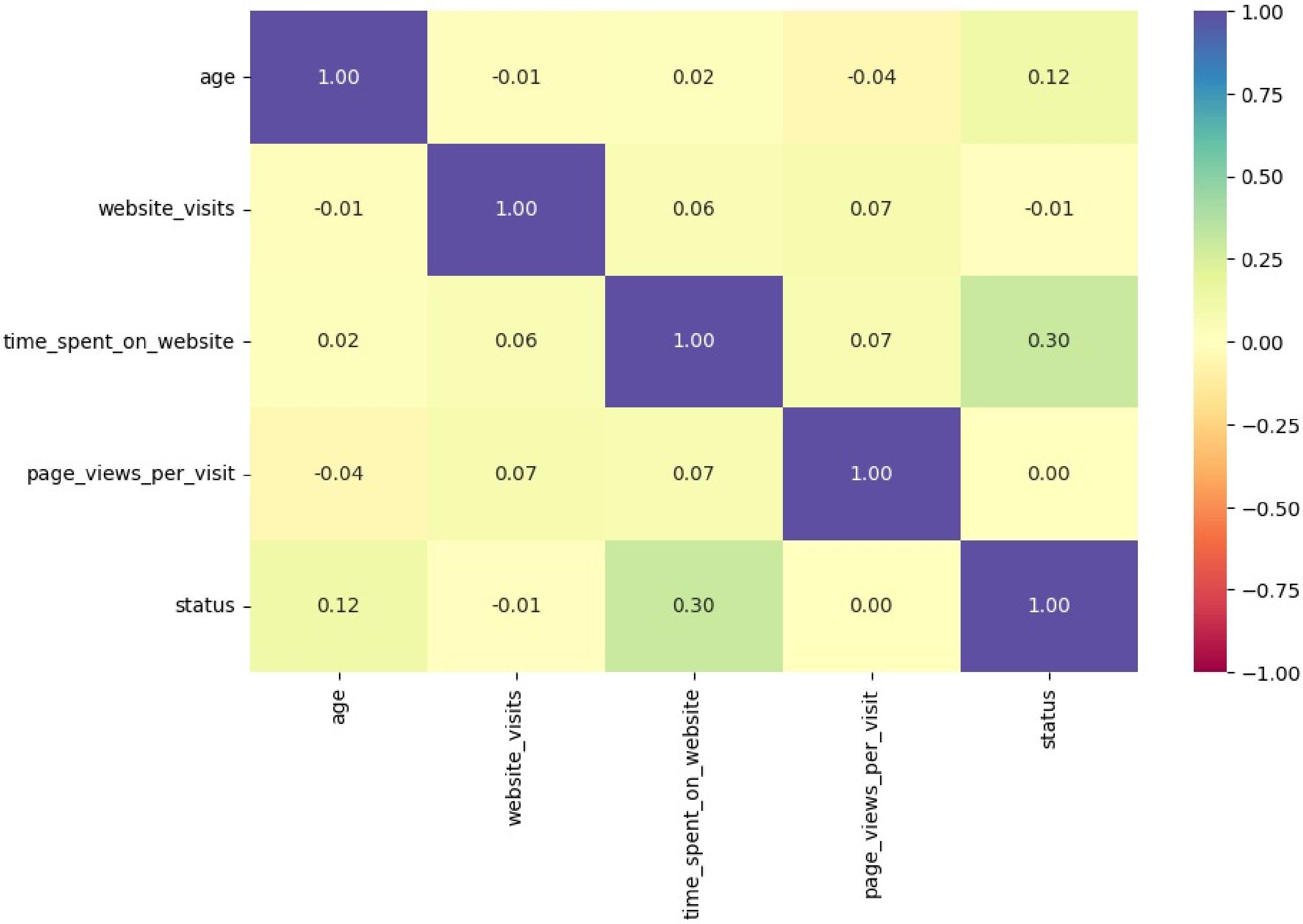
- Only about 30% of leads are converted into paying customers, indicating significant potential for improvement.
- Unconverted leads make up 70%, suggesting that the current marketing resources and strategies are not effective enough.

Bivariate Analysis

A heat map is used to check multicollinearity among variables.

In [208…

```
cols_list = data.select_dtypes(include=np.number).columns.tolist()

plt.figure(figsize=(10, 6))
sns.heatmap(
    data[cols_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectral"
)
plt.show()
```



Observations:

- Age has a has weak positive relationship with website_visits (0.12).
- Age and time_spent_on_website has a very weak negative relationship (-0.04).
- As majority of correlation coefficients are very low, it implies that the multicollinearity among variables are negligible.

Analysis of relationships between Features and Target

In order to find out how the features affect the status, we use stacked barpot and distribution plot to visualize the relationship between a categorical variable and the target variable (status). Staked barplot shows the proportion of features in different target classes (converted/not converted), and distribution plot shows how different features distribute across different target classes. This can help identify which groups are more likely to convert.

In this case, the codes for stacked barplot and distribution plot are reutilized from Low Code Version - Potential Customers Prediction. There are slight modifications to original codes:

1. Adjust the plot size for better readability
2. Add percentage figures to stacked barplot
3. Add green line (mean) and black line (median) to distribution plot

Numerical variables will be analyzed by stacked barplot, and categorical variables by histogram and boxplot.

In [209…

```
### function to plot distributions wrt target
def distribution_plot_wrt_target(data, predictor, target):
    fig, axs = plt.subplots(2, 2, figsize=(10, 7))
    target_uniq = data[target].unique()

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

    # Calculate mean and median
    mean_0 = data[data[target] == target_uniq[0]][predictor].mean()
    median_0 = data[data[target] == target_uniq[0]][predictor].median()

    # Add vertical lines for mean and median
    axs[0, 0].axvline(mean_0, color="green", linestyle="--", label='Mean')
    axs[0, 0].axvline(median_0, color="black", linestyle="-", label='Median')
    axs[0, 0].legend()

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

    # Calculate mean and median
    mean_1 = data[data[target] == target_uniq[1]][predictor].mean()
    median_1 = data[data[target] == target_uniq[1]][predictor].median()

    # Add vertical lines for mean and median
    axs[0, 1].axvline(mean_1, color="green", linestyle="--", label='Mean')
    axs[0, 1].axvline(median_1, color="black", linestyle="-", label='Median')
    axs[0, 1].legend()

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

    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",
    )
```

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

In [210…
```python
### function to stacked bar plot
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]

    # Create a cross-tabulation of predictor and target
    tab1 = pd.crosstab(data[predictor], data[target], margins=True).sort_values(
        by=sorter, ascending=False
    )

    # Calculate and format the Conversion Rate as a percentage string
    tab1['Conversion Rate'] = (tab1[1] / tab1['All']) * 100  # Assuming '1' is the paid customer status
    tab1['Conversion Rate'] = tab1['Conversion Rate'].map('{:.2f}%'.format)

    # Bold the "Conversion Rate" column header
    tab1_styled = tab1.style.set_table_styles(
        {
            'Conversion Rate': [{'selector': 'th', 'props': [('font-weight', 'bold')]}],
            '': [{'selector': 'th', 'props': [('font-weight', 'normal')]}],  # Other headers normal
        }
    )

    # Display the styled DataFrame
    display(tab1_styled)
    print("-" * 120)

    # Create normalized cross-tabulation
    tab = pd.crosstab(data[predictor], data[target], normalize="index").sort_values(by=sorter, ascending=False)

    # Plot the stacked bar chart
    ax = tab.plot(kind="bar", stacked=True, figsize=((count + 5) * 0.6, 5 * 0.8))

    # Annotate bars with percentage values
    for p in ax.patches:
        height = p.get_height()
        if height > 0:  # Avoid division by zero
            ax.annotate(f'{height:.2%}',
                        (p.get_x() + p.get_width() / 2., p.get_y() + height - 0.05),
                        ha='center', va='bottom', fontsize=9, color='black')

    plt.xticks(rotation=45, fontsize=9)
    plt.yticks(fontsize=9)
    plt.legend(loc="lower left", frameon=False)
    plt.legend(loc="upper left", bbox_to_anchor=(1, 1))
    plt.show()
```

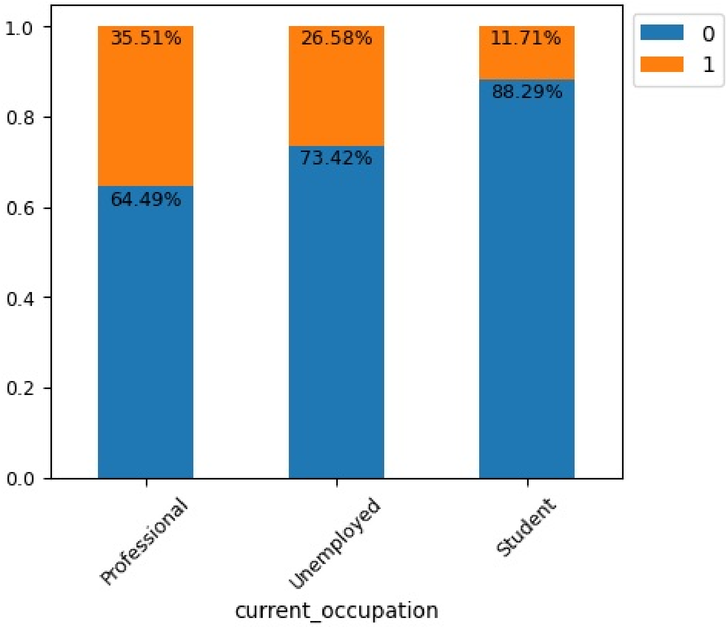
In [211…
```python
# calculate the interquartile ranges and mean
def getIQR(data, feature):
    """
    Calculate the interquartile range (Q1, Q2, Q3) and mean for a specified feature, broken down by status.

    Parameters:
    - data: pd.DataFrame: The input DataFrame containing the data.
    - feature: str: The name of the feature for which to calculate the IQR and mean.

    Returns:
    - list of str: A list containing formatted strings with the feature, status, IQR, and mean.
    """
    results = []

    # Get unique statuses
    statuses = data['status'].unique()

    print(f"Feature: {feature}")
    for status in statuses:
        # Filter the data based on the current status
        filtered_data = data[data['status'] == status][feature]

        # Calculate Q1, Q2 (median), and Q3
        Q1 = filtered_data.quantile(0.25)
        Q2 = filtered_data.median()  # Median is Q2
        Q3 = filtered_data.quantile(0.75)

        # Calculate mean
```

```
        mean_value = filtered_data.mean()

        # Format the result with two decimal places
        print('')
        print(f"Status: {status}")
        print(f"IQR: (Q1: {Q1:.2f}, Q2: {Q2:.2f}, Q3: {Q3:.2f})")
        print(f"Mean: {mean_value:.2f}")
```

Proportion of Conversions by Current Occupation

In [212…
```
stacked_barplot(data, "current_occupation", "status")
```

status current_occupation	0	1	All	Conversion Rate
All	3235	1377	4612	29.86%
Professional	1687	929	2616	35.51%
Unemployed	1058	383	1441	26.58%
Student	490	65	555	11.71%

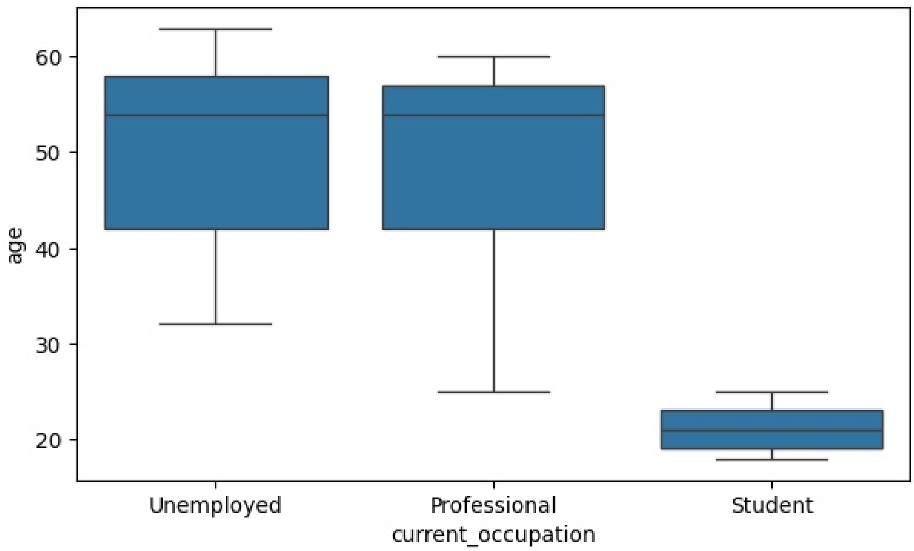


Observations:

- Professional has highest conversion rate, followed by unemployed and student.

Age Distribution by Current Occupation

In [213…
```
plt.figure(figsize=(7, 4))
sns.boxplot(data = data, x = data["current_occupation"], y = data["age"])
plt.show()
```



In [214…
```
data.groupby(["current_occupation"])["age"].describe()
```

Out[214…

current_occupation	count	mean	std	min	25%	50%	75%	max
Professional	2616.00000	49.34748	9.89074	25.00000	42.00000	54.00000	57.00000	60.00000
Student	555.00000	21.14414	2.00111	18.00000	19.00000	21.00000	23.00000	25.00000
Unemployed	1441.00000	50.14018	9.99950	32.00000	42.00000	54.00000	58.00000	63.00000

Observations:

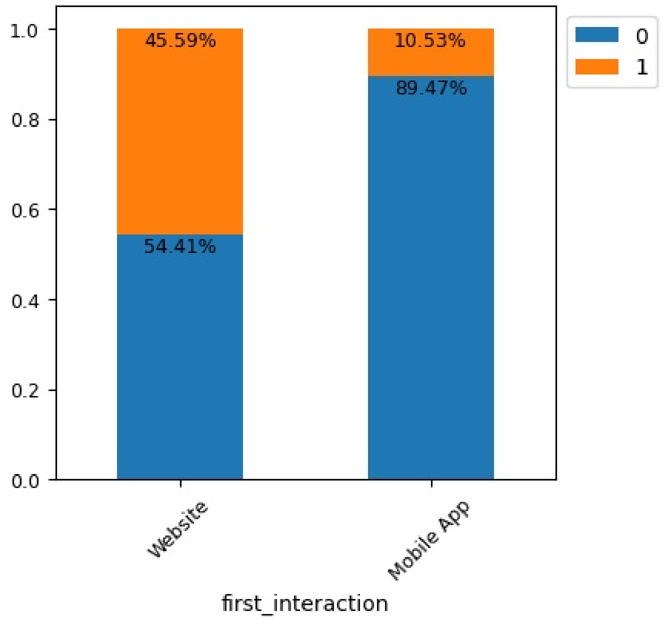
- Both Professional and unemployed have similar IQR of age 42 to 58, and also average age of around 50. Their max ages are 60-63.
- Student has aIQR of age at around 19-21, with a max age 25.

Proportion of Conversions by First Interaction

In [215…
```
stacked_barplot(data, "first_interaction", "status")
```

first_interaction \ status	0	1	All	Conversion Rate
All	3235	1377	4612	29.86%
Website	1383	1159	2542	45.59%
Mobile App	1852	218	2070	10.53%

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

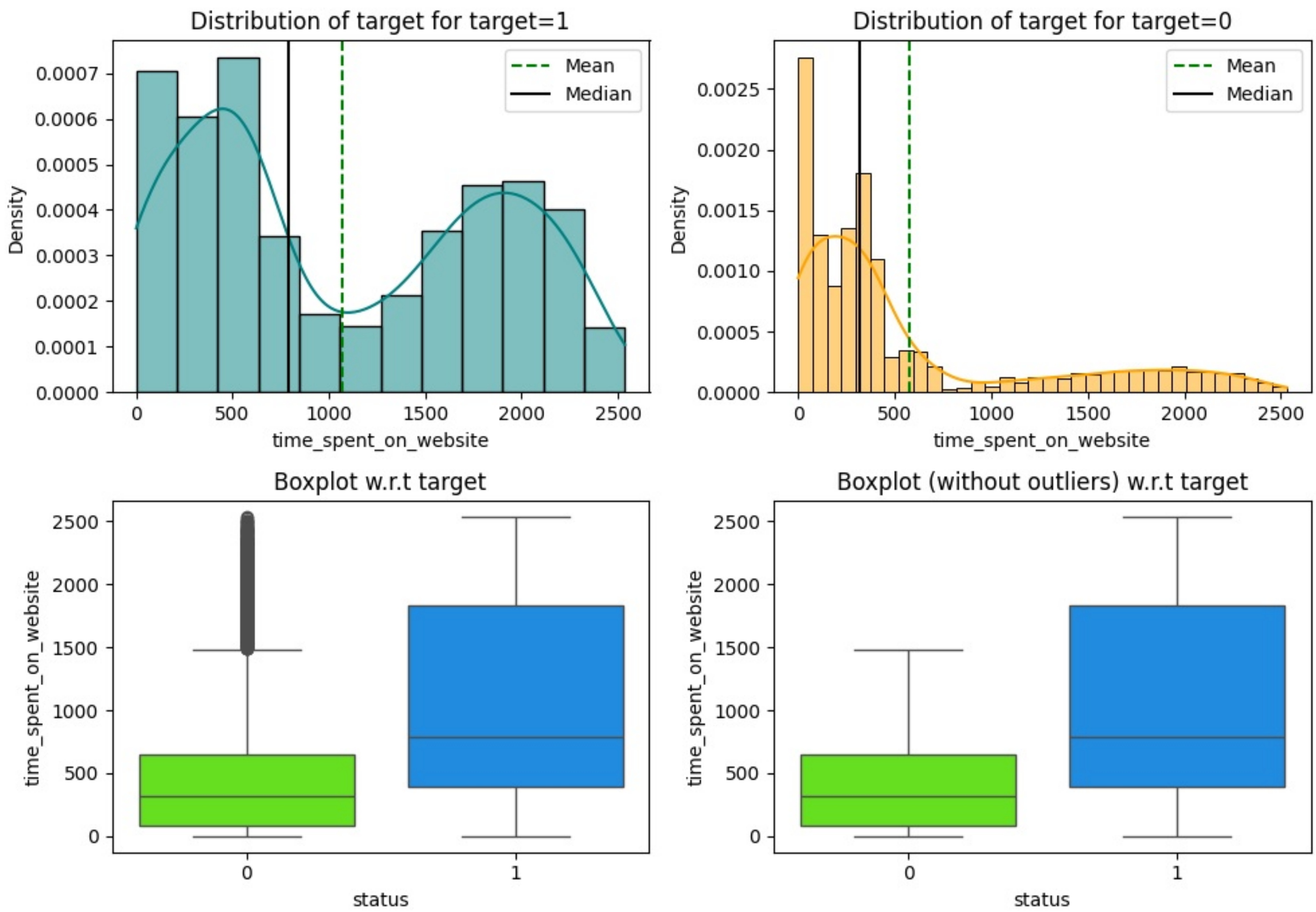


Observations:

- Website channel is much more effective in leads conversion than mobile app.

Analysis of Time Spent on Website by Conversion Status

In [216…
```
distribution_plot_wrt_target(data, "time_spent_on_website", "status")
```



Interquartile Range of time_spent_on_website across different classes (status=0 or 1)

In [217…
```
getIQR(data, "time_spent_on_website")
```

```
Feature: time_spent_on_website

Status: 1
IQR: (Q1: 390.00, Q2: 789.00, Q3: 1829.00)
Mean: 1068.40

Status: 0
IQR: (Q1: 88.00, Q2: 317.00, Q3: 646.00)
Mean: 577.42
```

Observations:

Converted Case (target=1)

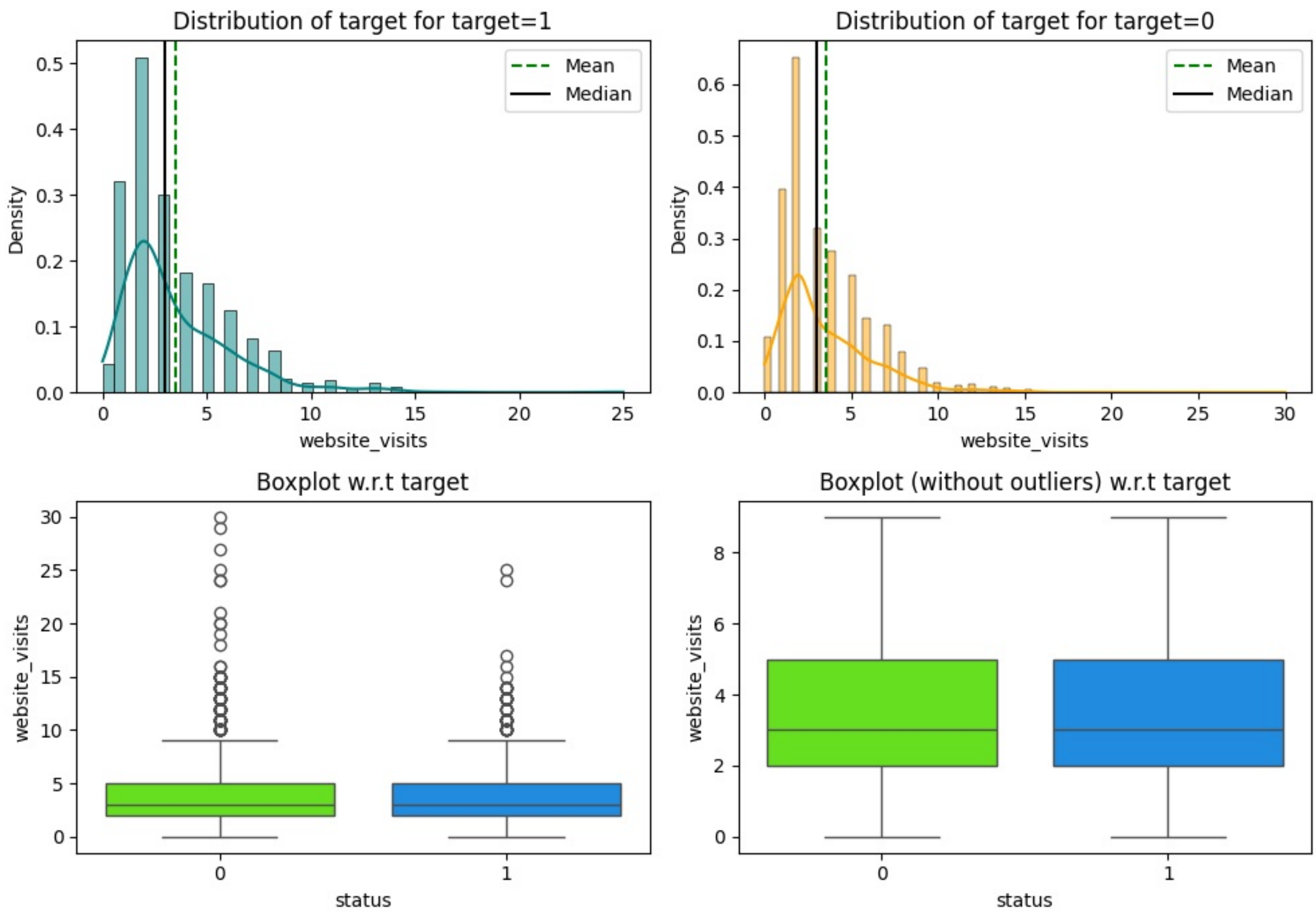
- The distribution plot shows a diversified distribution with peaks at around the 500 and 2000, indicating that these times are most common in converted group.
- Converted leads generally have 1068 average time spent on website.
- The boxplot shows a wider interquartile range (IQR) (390, 1829) compared to non-converted case, indicating greater variability in time in converted group.

Non-converted Case (target=0)

- The distribution plot shows that time spent on website is more concentrated at around lower values, indicating that non-converted users generally spent less time on the website.
- Non-converted group have a mean 577 time spent on website, which much lower than converted group.
- The boxplot shows that significant outliers are present in the non-converted group, indicating some users spent an exceptionally long time on the website but finally decide not to apply any course.

Analysis of Website Visits by Conversion Status

In [218…
```
distribution_plot_wrt_target(data, "website_visits", "status")
```



Interquartile Range

In [219…

```
getIQR(data, "website_visits")
```

```
Feature: website_visits

Status: 1
IQR: (Q1: 2.00, Q2: 3.00, Q3: 5.00)
Mean: 3.54

Status: 0
IQR: (Q1: 2.00, Q2: 3.00, Q3: 5.00)
Mean: 3.58
```

Observations:

Converted groups (target=1)

- The distribution plot is right-skewed with a mean 3.54 website visits. The website visit drops rapidly after 5 visits.
- Converted group have a IQR 2-5 and mean 3.54 website visits.
- Some converted leads have extra high website visits of 10 to over 15, and even 25.

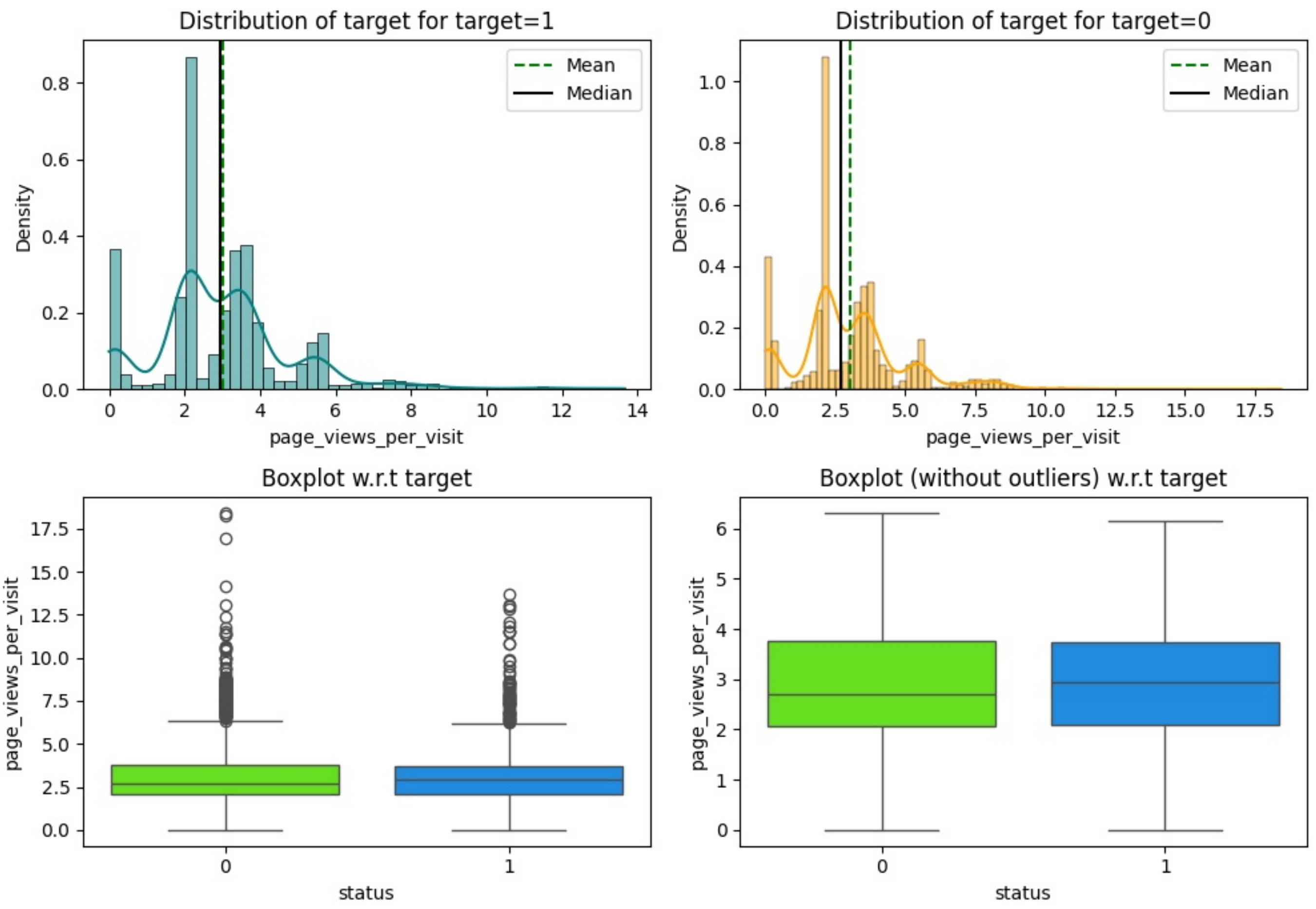
Non-converted Case (target=0)

- The distribution plot shape is similar to converted group.
- Non-converted group have a similar IQR and mean to converted group.
- Some non-converted leads have extra high website vists ranging from 10 to 30. This is noticeable because it means those leads decide not to join any courses even they're so many website visits.

Analysis of Page Views Per Visit by Conversion Status

In [220…

```
distribution_plot_wrt_target(data, "page_views_per_visit", "status")
```



Interquartile Range

In [221…
```
getIQR(data, "page_views_per_visit")
```

```
Feature: page_views_per_visit

Status: 1
IQR: (Q1: 2.08, Q2: 2.94, Q3: 3.73)
Mean: 3.03

Status: 0
IQR: (Q1: 2.07, Q2: 2.71, Q3: 3.77)
Mean: 3.03
```

Obseverations:

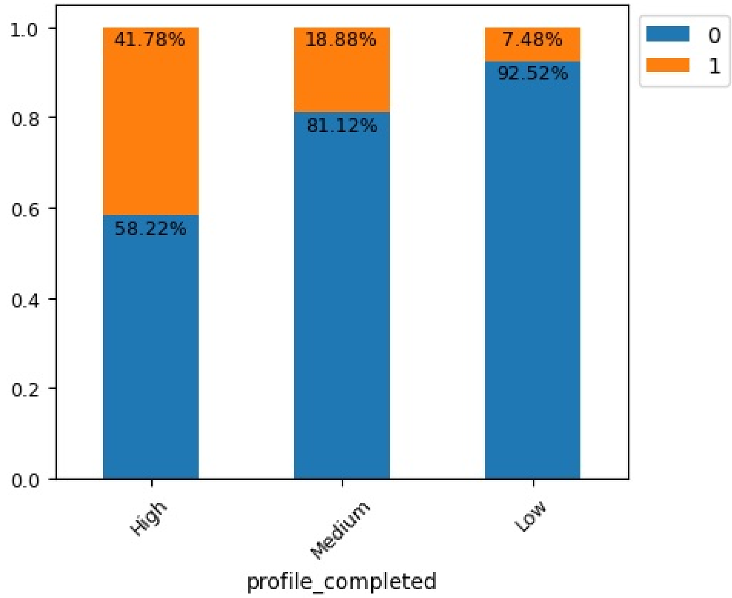
Converted / Non-converted groups

- Both the groups have similar distribution and mean.
- Converted group have average 2.94 pages per visit, while non-converted group is 2.74.
- The IQR of them is similar: 2.07-3.77
- Both converted / non-converted leads have extra high page views with range 6 to 13.
- Few non-converted leads even have 17.5 page views per visit.

Proportion of Conversions by Profile Completed

In [222…
```
stacked_barplot(data, "profile_completed", "status")
```

status	0	1	All	Conversion Rate
profile_completed				
All	3235	1377	4612	29.86%
High	1318	946	2264	41.78%
Medium	1818	423	2241	18.88%
Low	99	8	107	7.48%



Observations:

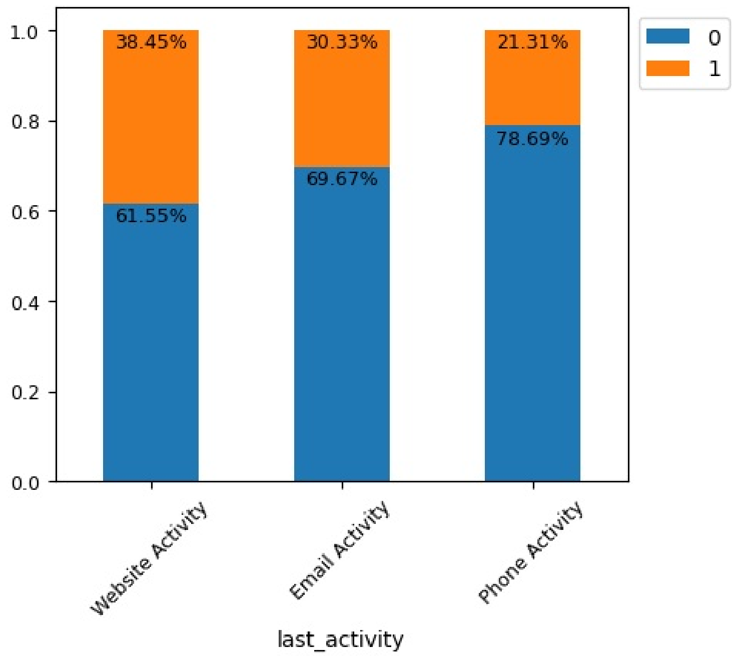
- High or medium profile completion have high conversion rate, followed by low level.

Proportion of Conversions by Last Activity

In [223…

```
stacked_barplot(data, "last_activity", "status")
```

status	0	1	All	Conversion Rate
last_activity				
All	3235	1377	4612	29.86%
Email Activity	1587	691	2278	30.33%
Website Activity	677	423	1100	38.45%
Phone Activity	971	263	1234	21.31%



Observations:

- Website activity or email activity have high conversion rate, followed by phone activity.

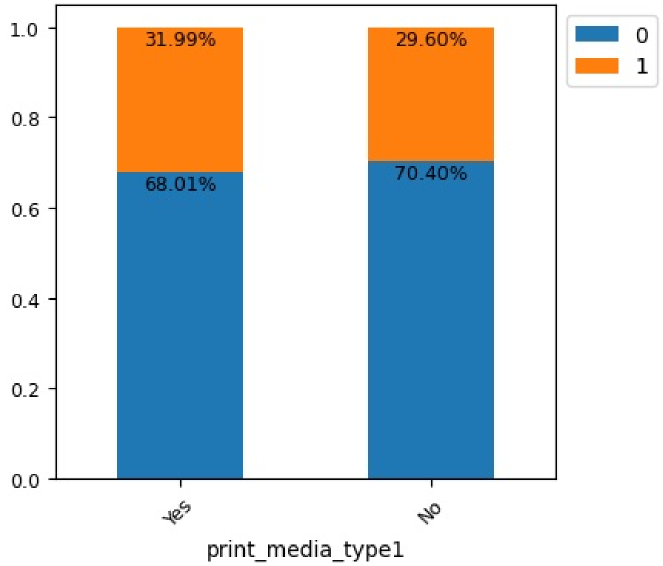
Proportion of Conversions by Print Media Type 1 (Newspaper Ad)

In [224…

```
stacked_barplot(data, "print_media_type1", "status")
```

status	0	1	All	Conversion Rate
print_media_type1				
All	3235	1377	4612	29.86%
No	2897	1218	4115	29.60%
Yes	338	159	497	31.99%

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



Observations:

- The conversion rates are similar in both group (Yes/No).
- Combined with the fact that only 10% leads access newspaper ad, it can conclude that newspaper ad doesn't have clear effect on conversion rate.

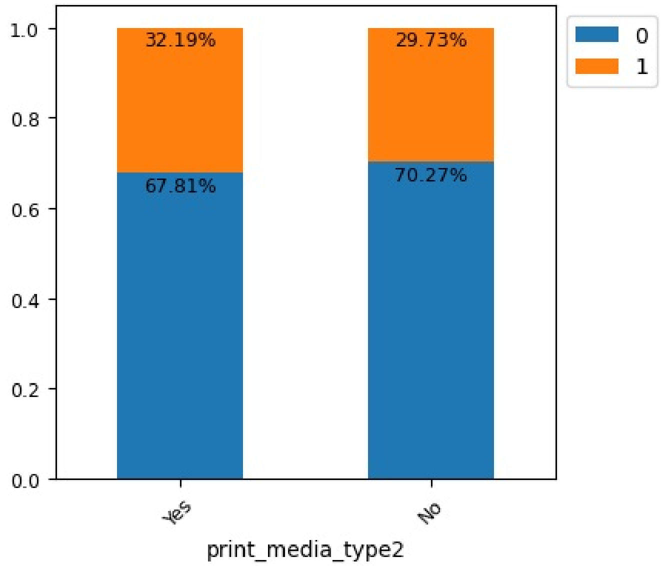
Proportion of Conversions by Print Media Type 2 (Magazine Ad)

In [225…

```
stacked_barplot(data, "print_media_type2", "status")
```

status	0	1	All	Conversion Rate
print_media_type2				
All	3235	1377	4612	29.86%
No	3077	1302	4379	29.73%
Yes	158	75	233	32.19%

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



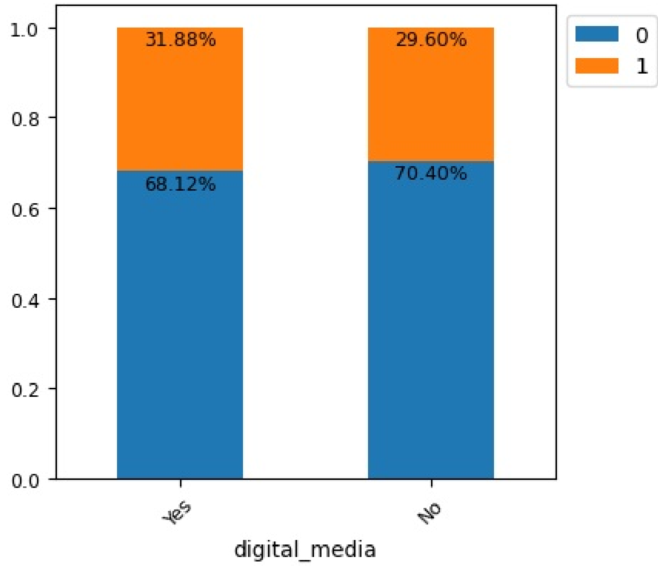
Observations:

- The conversion rates are similar in both group (Yes/No).
- Combined with the fact that only 5% leads access magazine ad, it can conclude that magazine ad doesn't have clear effect on conversion rate.

Proportion of Conversions by Digital Media

In [226…
```
stacked_barplot(data, "digital_media", "status")
```

status	0	1	All	Conversion Rate
digital_media				
All	3235	1377	4612	29.86%
No	2876	1209	4085	29.60%
Yes	359	168	527	31.88%



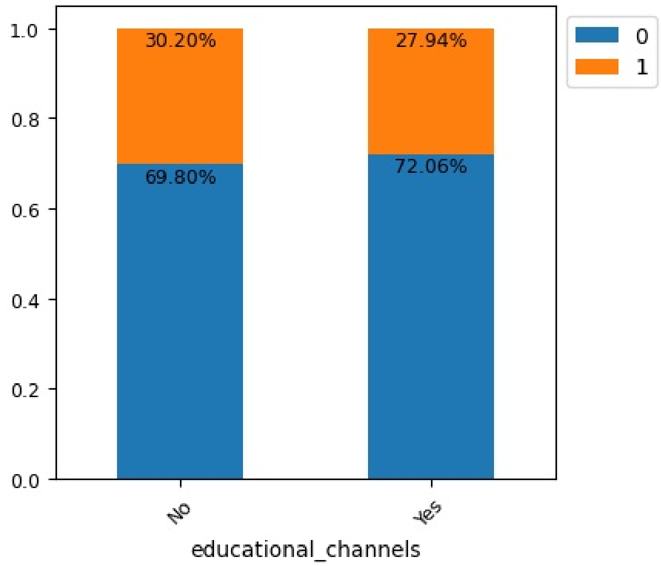
Observations:

- Similar to above.

Proportion of Conversions by Educational Channels

In [227…
```
stacked_barplot(data, "educational_channels", "status")
```

status	0	1	All	Conversion Rate
educational_channels				
All	3235	1377	4612	29.86%
No	2727	1180	3907	30.20%
Yes	508	197	705	27.94%



Observations:

- Similar to above.

Proportion of Conversions by Referral

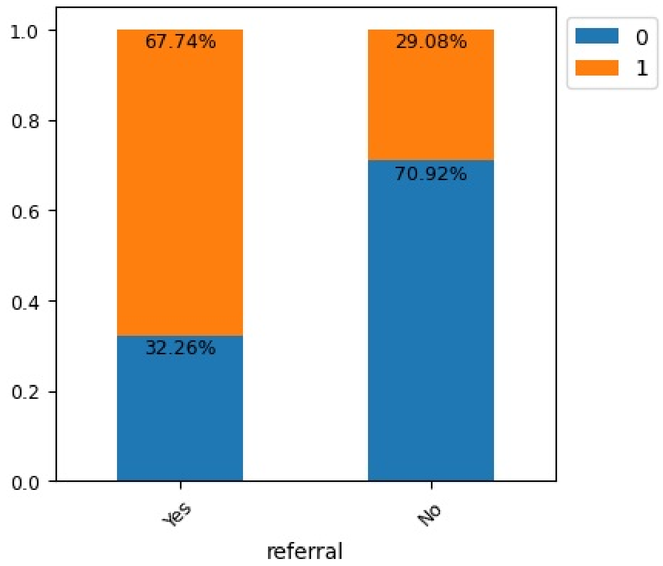
Observations:

- Similar to above.

In [228…

```
stacked_barplot(data, "referral", "status")
```

status	0	1	All	Conversion Rate
referral				
All	3235	1377	4612	29.86%
No	3205	1314	4519	29.08%
Yes	30	63	93	67.74%



Observations:

- Similar to above.

Data Preparation for modeling

- The most important business objective is to predict which lead is more likely to be converted.
- Categorical features should be encoded for modelling.
- Data will be split into training and testing.

In [229…

```
X = data.drop(["ID", "status"], axis=1) # independent variables
Y = data["status"] # dependent (target) variable
X = pd.get_dummies(X, drop_first=True) # dummies for X

# Splitting the data 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, random_state=1
)
```

In [230…

```
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, but the lead would have converted to a paid customer.
2. Predicting a lead will be converted to a paid customer in reality, but 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 be converted. This is a critical scenario as it indicates the model's failure to recognize potential customers. The company might miss out on nurturing these leads effectively, resulting in lost revenue.

- If we predict that a lead will get converted and the lead doesn't get converted. This scenario represents a missed opportunity for resource allocation effectively since the company may invest time and effort to nurture these leads unnecessarily. It could also lead to wasted marketing resources.

Losing a potential customer is a greater loss.

How to reduce the losses?

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

In [231…

```
# Remark: this function is reutilized from Low Code Version - Potential Customers Prediction
# Function to print the classification report and get confusion matrix in a proper format
def metrics_score(actual, predicted):
    print(classification_report(actual, predicted))
    cm = confusion_matrix(actual, predicted)
    plt.figure(figsize = (8, 5))
    sns.heatmap(cm, annot = True,  fmt = '.2f', xticklabels = ['Not Converted', 'Converted'], yticklabels = ['N
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()
```

Decision Tree

Building Decision Tree Model

In [232…

```
# Fitting the decision tree classifier on the training data
features = data.copy()
features = features.drop(["ID", "status"], axis=1)
d_tree = DecisionTreeClassifier(random_state=42)

d_tree.fit(X_train, y_train)
```

Out[232…

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

Checking the Decision Tree performance on training set

In [233…

```
# 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:

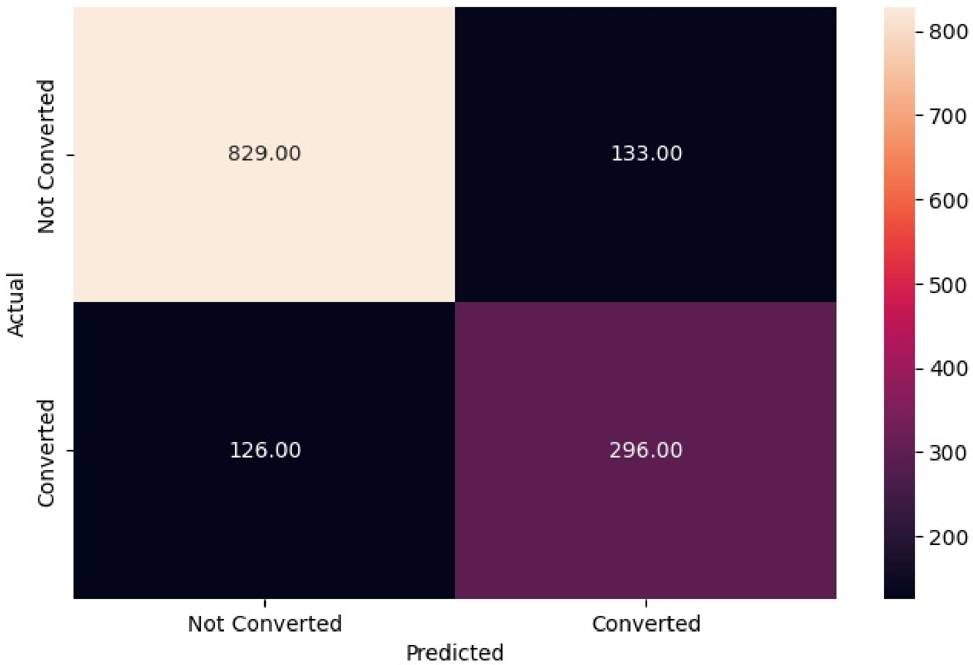
- The Decision tree is giving a **100% score for all metrics on the training dataset.**

Checking the Decision Tree performance on testing set

In [234…

```
# 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:

Overall Performance:

- The model achieves an overall accuracy of 81%
- Weighted average F1-score is 0.81, indicating good overall performance

Class-specific Performance:

Class 0 (Not Converted):

- Precision: 0.87 - When the model predicts a lead won't convert, it's correct 87% of the time
- Recall: 0.86 - The model correctly identifies 86% of all non-converting leads
- F1-score: 0.86 - The model shows balanced performance for this class
- 829 true negatives (correctly predicted non-conversions)
- 133 false negatives (incorrectly predicted as non-conversions)

Class 1 (Converted):

- Precision: 0.69 - When the model predicts a lead will convert, it's correct 69% of the time
- Recall: 0.70 - The model correctly identifies 70% of all converting leads
- F1-score: 0.70 - The model indicates room for improvement in identifying conversions
- 296 true positives (correctly predicted conversions)
- 126 false positives (incorrectly predicted as conversions)

Class Imbalance:

- The dataset is imbalanced with 962 non-converted leads (class 0) vs. 422 converted leads (class 1)
- This imbalance ratio of approximately 2.3:1 affects the model's performance on the minority class

Error Analysis:

- False Positives (126): The model incorrectly predicted conversion for these leads
- False Negatives (133): The model missed these potential conversions
- The model shows similar error rates in both directions, suggesting balanced misclassification

Business Implications:

- The model is better at identifying non-converting leads (86% recall) than converting leads (70% recall)
- This suggests the model could be more valuable for filtering out unlikely leads than for identifying high-potential leads
- The relatively high false negative rate (133 missed conversions) represents potential lost business opportunities

Potential Improvements:

- Implementing class weights to address the imbalance could improve recall for the minority class
- Feature engineering or additional data might help better distinguish between the classes
- Hyperparameter tuning could potentially improve overall performance

The model shows decent predictive power but has room for improvement, particularly in correctly identifying potential conversions.

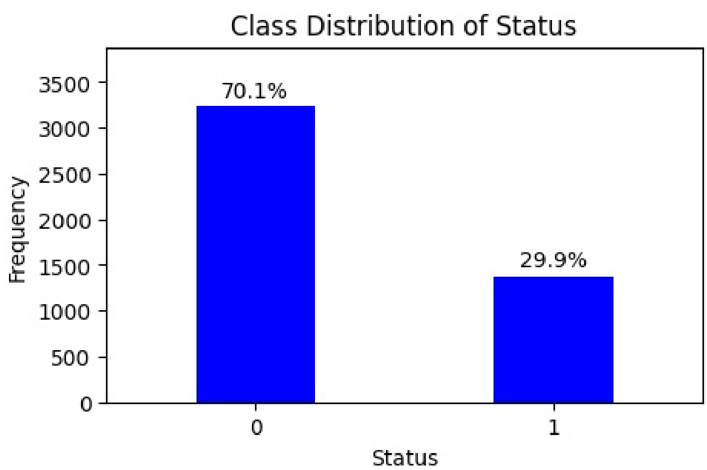
Decision Tree - Hyperparameter Tuning

THe following bar plot is used to determine whether a class_weight hyperparameter should be used to build the decision tree.

Determine if the dataset has imbalance

In [235…

```
# first determine whether a class weight is needed for a decision tree model
class_distribution = data['status'].value_counts()
total_count = class_distribution.sum()

# Calculate percentages
percentages = (class_distribution / total_count) * 100

# Plotting
plt.figure(figsize=(5, 3))
bars = plt.bar(class_distribution.index, class_distribution.values, color='blue', width=0.4)

# Adding percentage labels on top of the bars
for bar, percentage in zip(bars, percentages):
    yval = bar.get_height()
    #plt.text(bar.get_x() + bar.get_width()/2, yval, f'{percentage:.1f}%', ha='center', va='bottom')
    plt.text(bar.get_x() + bar.get_width()/2, yval + 50, f'{percentage:.1f}%', ha='center', va='bottom')  # Adj

# Plot aesthetics
plt.title('Class Distribution of Status')
plt.xlabel('Status')
plt.ylabel('Frequency')
plt.xticks([0, 1])  # Set x-ticks to 0 and 1
plt.xlim(-0.5, 1.5)  # Set limits to center the bars properly

# Adjust y-axis limits to create space at the top
plt.ylim(0, class_distribution.max() * 1.2)  # Adjusted ylim factor

plt.show()
```



Observations:

- Percentage of status 0 significantly exceeds those with status 1, indicating an imbalance in the dataset.
- Since one of the business objectives is to identify potential converted leads, greater importance should be assigned to the minority class.

- A class weights (0: 0.3, 1: 0.7) will be used when building the decision tree.

The decision tree will be built with class weights and other parameters for avoiding overfitting.

In [236…
```
# Choose the type of classifier
d_tree_tuned = DecisionTreeClassifier(random_state = 42, class_weight = {0: 0.3, 1: 0.7})

# 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 class 1
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[236…

```
DecisionTreeClassifier

DecisionTreeClassifier(class_weight={0: 0.3, 1: 0.7}, criterion='entropy',
                       max_depth=np.int64(3), min_samples_leaf=5,
                       random_state=42)
```

Checking the Optisimized Decision Tree performance on training set

In [237…
```
# Checking performance on the training data
y_pred_train2 = d_tree_tuned.predict(X_train)  # Predictions on training set with tuned model
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
```

Checking the Optisimized Decision Tree performance on testing set

In [238…
```
# Checking performance on the testing data
y_pred_test2 = d_tree_tuned.predict(X_test)
metrics_score(y_test, y_pred_test2)  # Evaluate the tuned model's performance
```

```
              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:

Based on testing data results, comparing the optimized decision tree with the untuned decision tree, here are the key observations:

1. Recall for Converted Class (Class 1):

- Optimized: 0.86 (86%)
- Untuned: 0.70 (70%)
- Improvement: +16% - This is a significant improvement in identifying actual converters, which was the optimization goal - (using recall as the scoring metric)

2. Precision Trade-off:

- Optimized: 0.62 for Class 1 (down from 0.69)
- Untuned: 0.69 for Class 1
- The model sacrificed some precision to achieve better recall

3. Confusion Matrix Changes:

- False Negatives: Decreased from 126 to 57 (fewer missed conversion opportunities)
- False Positives: Increased from 133 to 224 (more non-converters incorrectly identified as converters)
- True Positives: Increased from 296 to 365 (more actual converters correctly identified)

4. Overall Accuracy:

- Optimized: 0.80 (80%)
- Untuned: 0.81 (81%)
- Slight decrease in overall accuracy, but this is acceptable given the business goal

5. Class Weights Impact:

- The class_weight={0: 0.3, 1: 0.7} parameter successfully shifted the model's focus toward better identifying the minority class (converters)
- This is evident in the improved recall for Class 1

6. Business Implications:

- The optimized model would help the business identify 69 more potential customers (365 vs. 296)
- The cost is 91 more false positives (224 vs. 133), which means more resources spent on leads that won't convert
- This trade-off is often acceptable in lead conversion scenarios where missing potential customers is more costly than - spending resources on non-converters

7. F1-Score:

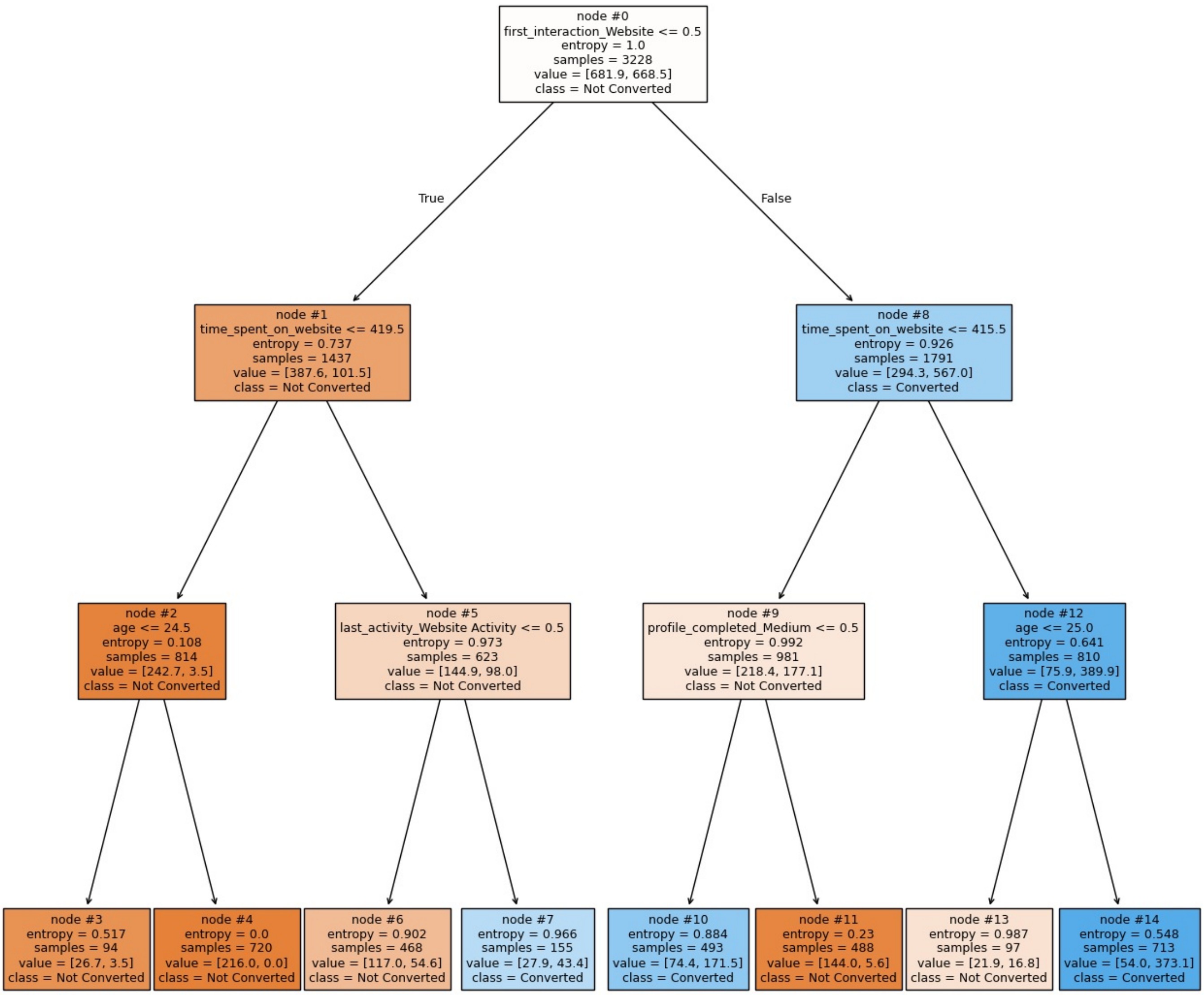
- Class 1 F1-score improved slightly from 0.70 to 0.72, showing a better balance between precision and recall for the conversion class

The optimization was successful in achieving its primary goal of improving recall for the converted class, which is typically the more important business objective in lead conversion scenarios.

Visualizing the Decision Tree

In [239…
```
features = list(X.columns)
print(features)
plt.figure(figsize = (16, 16))

# convert means the leads converted to paid customers
class_names = ['Not Converted', 'Converted']
tree.plot_tree(d_tree_tuned, feature_names = features, filled = True, fontsize = 9, node_ids = True, class_name:
plt.show()
print(d_tree_tuned)
```

```
['age', 'website_visits', 'time_spent_on_website', 'page_views_per_visit', 'current_occupation_Student', 'curren
t_occupation_Unemployed', 'first_interaction_Website', 'profile_completed_Low', 'profile_completed_Medium', 'las
t_activity_Phone Activity', 'last_activity_Website Activity', 'print_media_type1_Yes', 'print_media_type2_Yes',
'digital_media_Yes', 'educational_channels_Yes', 'referral_Yes']
```




```
DecisionTreeClassifier(class_weight={0: 0.3, 1: 0.7}, criterion='entropy',
                       max_depth=np.int64(3), min_samples_leaf=5,
                       random_state=42)
```

Observations:

Ranking list of factors on leads conversion:

1. First Interaction Method (Mobile vs Website):

- This is the root node split, indicating it's the most determinating feature.
2. Time Spent on Website:
   - Appears immediately after the root split.
   - Longer sessions generally correlate with higher conversion probability.
3. Age:
   - Multiple splits based on age thresholds (24.5 and 25.0).
   - Shows that different age groups have distinct conversion patterns.
4. Profile Completion Status
   - Indicates user investment of time in ExtraaLearn platform.
   - Shows commitment level correlates with conversion likelihood.
5. Last Activity Type
   - indicates that the nature of lead's last interaction can predict conversion.

The above ranking list is based on entropy values, node depth and sample distribution.

Characteristics of Converted Cases:

1. Path Node 1->Node 5->Node 7:
   - Node 1: More time spent on website implies that the leads have more interest in ExtraaLearn
   - Node 5: More website activities as last interaction implies that the leads may concentrate in browsing course information or payment details.
2. Path Node 8->Node 9->Node 10:
   - Node 8 & 9: though the time spent on website and profile completed are both lower than threshold values, but ultimately the leads are converted.
   - It may indicate that those leads may be the type of persons that are not very keen on social media but are still interested in online education.
3. Path Node 8->Node 12->Node 14:
   - Node 8: more time spent on website implies that the leads have more interest in ExtraaLearn
   - Node 12: age over 25 may suggest leads are likely employed and so they are more affordable to pay for online education.

Is Pruning of the Tuned Decision Tree needed?

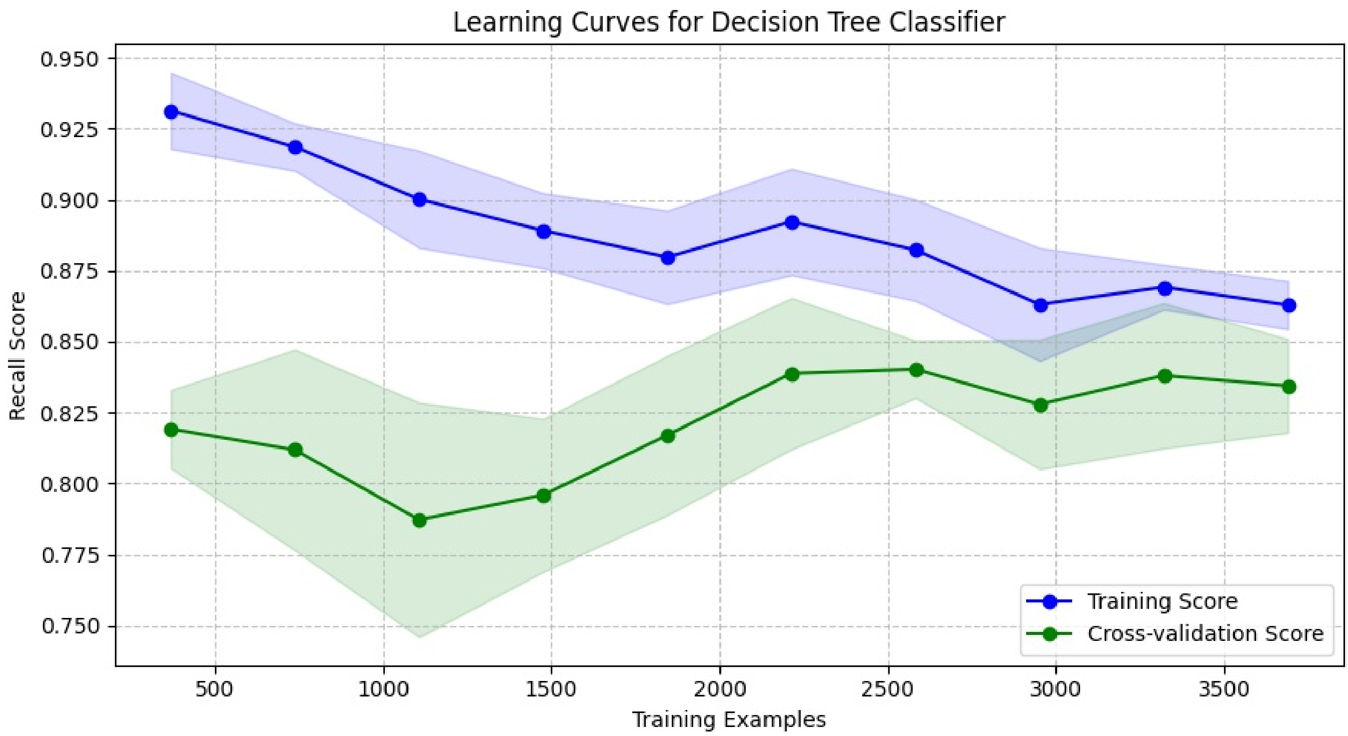
A learning curve is used to find out the answer.

In [240…

```
# Prepare features and target
X = data.drop(['status', 'ID'], axis=1)
y = data['status']

# Convert categorical variables
categorical_features = ['current_occupation', 'first_interaction', 'profile_completed', 'last_activity']
X = pd.get_dummies(X, columns=categorical_features)

# Convert boolean columns
boolean_columns = ['print_media_type1', 'print_media_type2', 'digital_media', 'educational_channels', 'referral
for col in boolean_columns:
    X[col] = X[col].map({'Yes': 1, 'No': 0})

# Initialize the model with best parameters from grid search
d_tree = DecisionTreeClassifier(
    random_state=42,
    class_weight={0: 0.3, 1: 0.7},
    criterion='entropy',
    max_depth=6,
    min_samples_leaf=10
)

# Calculate learning curve
train_sizes, train_scores, val_scores = learning_curve(
    d_tree, X, y,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=5,
    scoring='recall',
    n_jobs=-1
)

# Calculate mean and standard deviation
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
val_mean = np.mean(val_scores, axis=1)
val_std = np.std(val_scores, axis=1)

# Create the learning curve plot
plt.figure(figsize=(9, 5))
plt.grid()
```

```python
# Plot training scores
plt.plot(train_sizes, train_mean, label='Training Score', color='blue', marker='o')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.15, color='blue')

# Plot cross-validation scores
plt.plot(train_sizes, val_mean, label='Cross-validation Score', color='green', marker='o')
plt.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, alpha=0.15, color='green')

# Customize the plot
plt.xlabel('Training Examples', fontsize=10)
plt.ylabel('Recall Score', fontsize=10)
plt.title('Learning Curves for Decision Tree Classifier', fontsize=12)
plt.legend(loc='lower right', fontsize=10)

# Add grid for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Print numerical results
print("\nLearning Curve Analysis:")
print("-" * 50)
print(f"Final Training Score: {train_mean[-1]:.3f} (±{train_std[-1]:.3f})")
print(f"Final Validation Score: {val_mean[-1]:.3f} (±{val_std[-1]:.3f})")
print(f"Gap between Training and Validation: {train_mean[-1] - val_mean[-1]:.3f}")

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

```
Learning Curve Analysis:
--------------------------------------------------
Final Training Score: 0.863 (±0.009)
Final Validation Score: 0.834 (±0.016)
Gap between Training and Validation: 0.028
```



Observations:

- Both curves plateau after training set size 3000, suggesting more data won't significantly improve performance.
- Both training and cross-validation line converges gradually as the training set size increases, suggesting the model has found a good balance between bias and variance.

Conclusions:

- This learning curve implies that the current pruning parameters are appropriate and additional pruning is not needed.

Feature importance of the tuned decision tree model

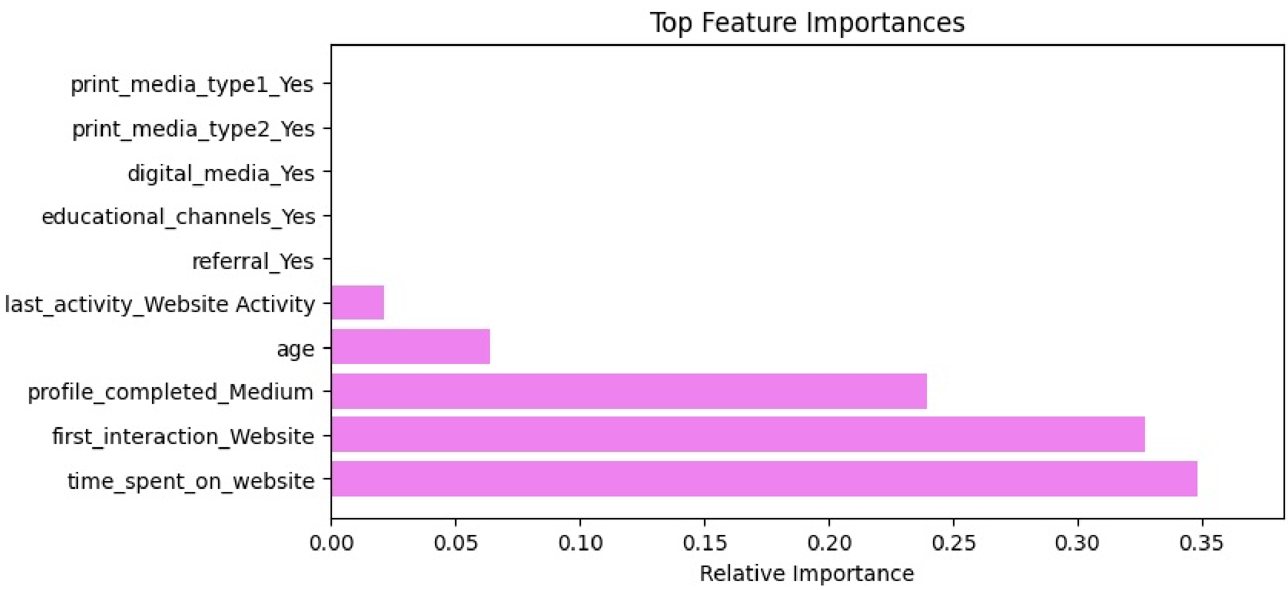
In [241…

```python
# Importance of features in the tree building
print (pd.DataFrame(d_tree_tuned.feature_importances_, columns = ["Imp"], index = X_train.columns).sort_values(I
```

```
                                     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 [242…

```
# Plotting the feature importance
importances = d_tree_tuned.feature_importances_
indices = np.argsort(importances)[::-1]  # Sort in descending order
# Limit to top N features
top_n = 10
top_indices = indices[:top_n]
plt.figure(figsize=(8, 4))
plt.title('Top Feature Importances')
plt.barh(range(top_n), importances[top_indices], color='violet', align='center')
plt.yticks(range(top_n), [X_train.columns[i] for i in top_indices], fontsize=10)
plt.xticks(fontsize=10)
plt.xlabel('Relative Importance', fontsize=10)
plt.xlim(0, max(importances[top_indices]) * 1.1)  # Adjust x-axis limit
plt.show()
```



Observations:

- Time spent on the website and first_interaction_website are the most important features, followed by profile_completed, age, and last_activity.
- The rest of the variables have no impact in this model, while deciding whether a lead will be converted or not.

Random Forest Classifier

Building Random Forest Model

In [243…

```
# Fitting the random forest tree classifier on the training data
rf_estimator = RandomForestClassifier(random_state=42)
rf_estimator.fit(X_train, y_train)
```

Out[243…

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

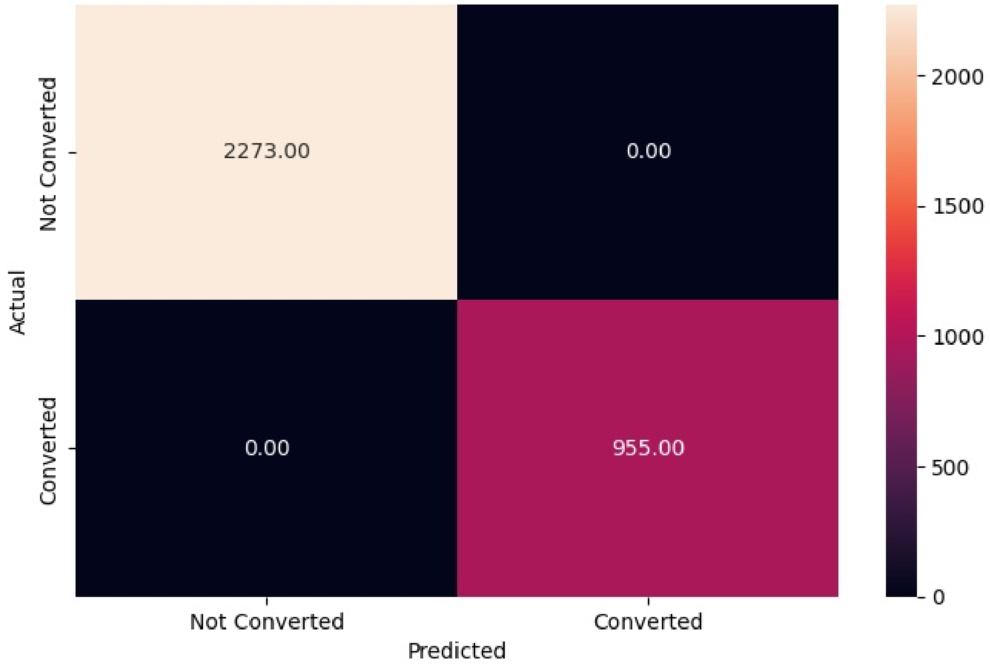
Checking the Random Forset model performance on training set

In [244…

```
# 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:

- The Decision tree is giving a **100% score for all metrics on the training dataset.**

Checking the Random Forset model performance on testing set

In [245…

```
# 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      0.90       962
           1       0.80      0.69      0.74       422

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

Observations:

1. Overall Performance:

- 85% accuracy - strong overall performance
- Balanced metrics across precision and recall

2. Predicting Non-Converters (Class 0):

- 92% recall - excellent at identifying non-converting leads
- 87% precision - high reliability when predicting someone won't convert Only 75 false positives - minimal wasted resources

3. Predicting Converters (Class 1):

- 69% recall - identifies about 7 out of 10 actual converters
- 80% precision - when it predicts conversion, it's right 80% of the time
- 131 false negatives - missing some potential conversion opportunities

4. Business Impact:

- Strong at avoiding false positives (predicting conversion when it won't happen)
- Moderate at capturing all potential converters
- Good balance between precision and recall for business decision-making

5. Confusion Matrix Highlights:

- 887 true negatives - correctly identified non-converters
- 291 true positives - correctly identified converters
- Overall shows good classification performance across both classes

This untuned Random Forest performs well out-of-the-box, with particularly strong performance on identifying non-converters while maintaining good precision for the conversion class.

Random Forest Classifier - Hyperparameter Tuning

The following parameters are used to tune the decision tree:

- n_estimators: Find the optimal number of trees can help balance performance and efficiency
- max_depth: Helps control the complexity of the model, making it more generalizable to unseen data.
- min_samples_leaf: Reduces overfitting by ensuring that leaves have a minimum number of observations
- max_features: Helps to reduce overfitting and allows the model to generalize better by using a subset of features
- max_samples: Encourages diversity among the trees, which can lead to better model performance.
- class_weight: Useful for handling imbalanced datasets, ensuring that the model pays appropriate attention to all classes, especially the minority class

In [252…

```
# Choose the type of classifier
rf_estimator_tuned = RandomForestClassifier(criterion = "entropy", random_state = 7)

# 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 class 1
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(estimator=rf_estimator_tuned, param_grid=parameters, scoring=scorer, cv=5)

# Fit the grid search to the data
grid_obj.fit(X_train, y_train)

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

In [253…

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

Out[253…

```
RandomForestClassifier

RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=6, max_features=0.8, max_samples=0.9,
                       min_samples_leaf=25, n_estimators=120, random_state=7)
```

Checking the Optimized Random Forset model performance on training set

In [254…

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

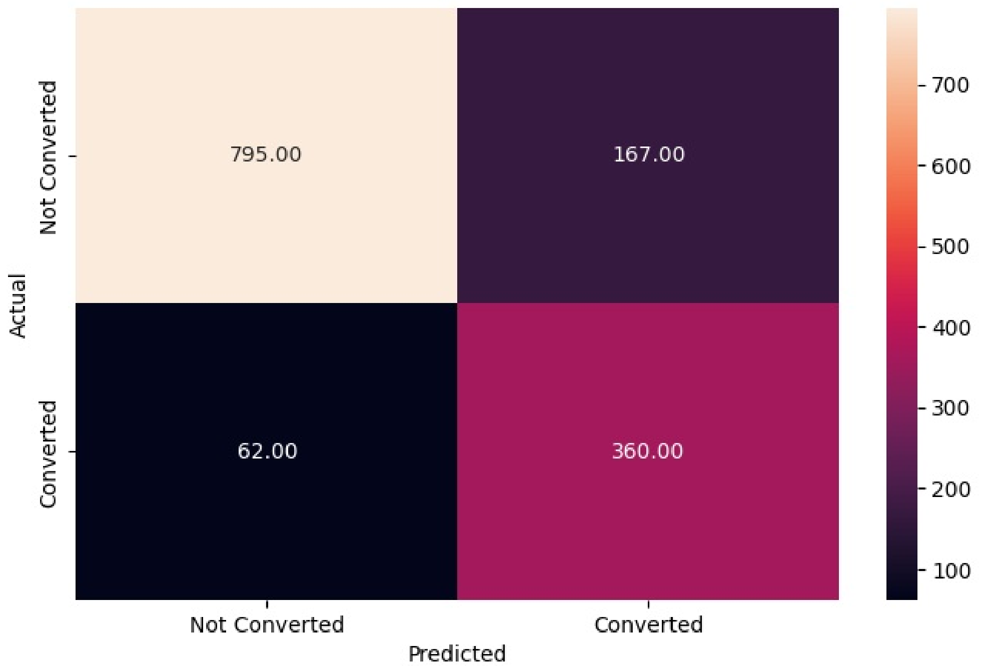
```
              precision    recall  f1-score   support

           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
```

Checking the Optimized Random Forset model performance on testing set

In [255…

```
# Checking performance on the test data
y_pred_train4 = rf_estimator_tuned.predict(X_test)
metrics_score(y_test, y_pred_train4)
```

```
              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:

Comparing the optimized Random Forest with the untuned version, here are the key observations:

1. Overall Performance Improvements

- Accuracy: Changed from 85% to 83% (slight decrease)
- Weighted Average F1-Score: Changed from 0.85 to 0.84 (slight decrease)

2. Class 0 (Not Converted) Performance

- Precision: Improved from 0.87 to 0.93 (+6%)
- Recall: Decreased from 0.92 to 0.83 (-9%)
- True Negatives: Decreased from 887 to 795 cases

3. Class 1 (Converted) Performance

- Precision: Decreased from 0.80 to 0.68 (-12%)
- Recall: Significantly improved from 0.69 to 0.85 (+16%)
- True Positives: Increased from 291 to 360 cases (+69 more conversions identified)

4. Error Distribution Changes

- False Negatives: Decreased from 131 to 62 (-69 cases)
- False Positives: Increased from 75 to 167 (+92 cases)

5. Business Impact Analysis

- Better at Finding Conversions: The optimized model identifies 16% more actual conversions (higher recall for class 1)
- More False Alarms: The model now incorrectly flags more non-converters as converters
- Trade-off Shift: Clear shift from precision to recall for the conversion class

6. Strategic Implications

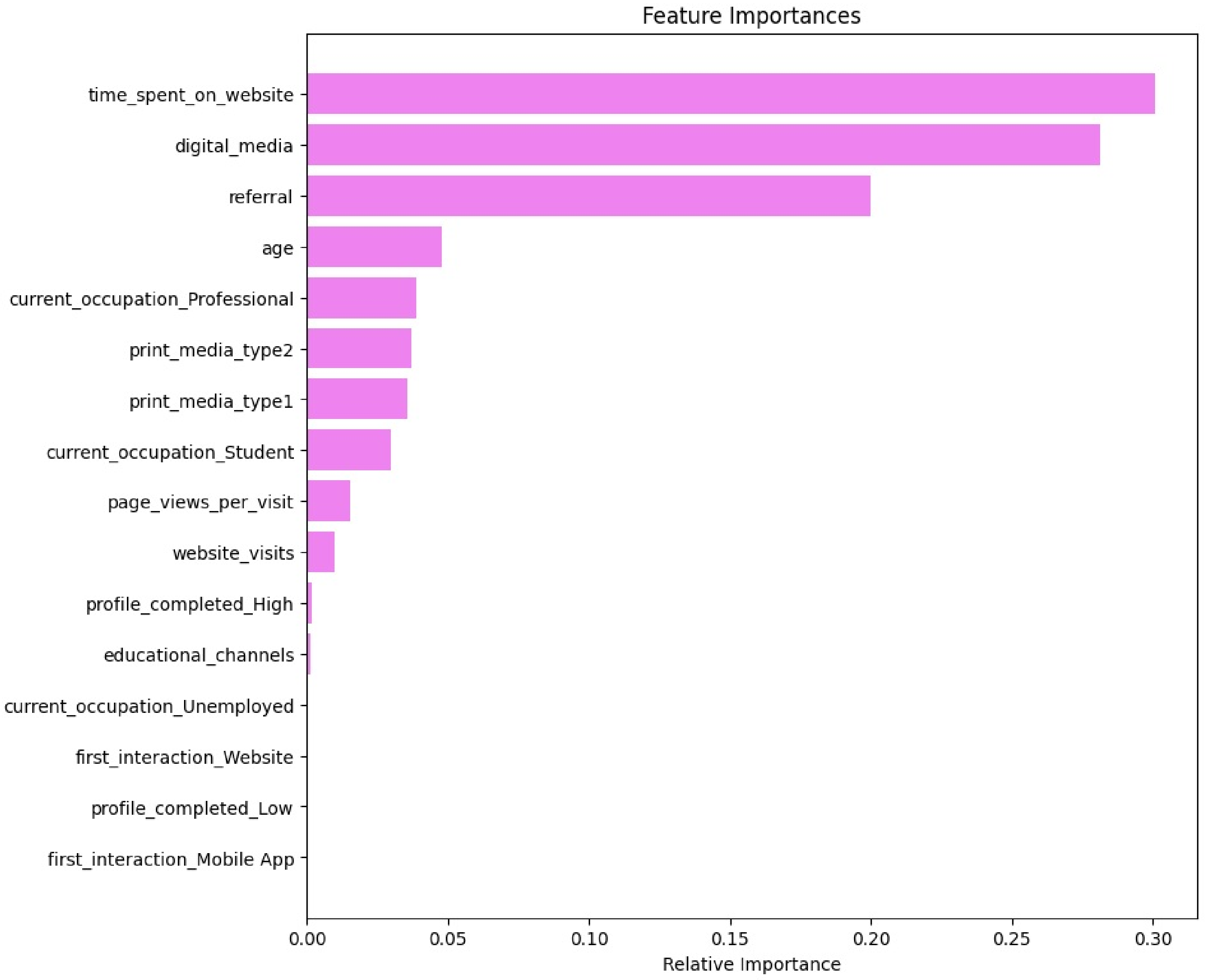
- The optimized model is better suited for applications where:
- Finding all potential converters is more important than precision
- Missing conversion opportunities is considered more costly than pursuing false leads
- Marketing resources are available to handle more potential leads, even if some won't convert

The optimization has effectively shifted the model's focus toward identifying more potential conversions at the cost of some precision, which may align better with business goals of maximizing conversion opportunities.

Feature Importance of the model

In [256…

```
importances = rf_estimator_tuned.feature_importances_
indices = np.argsort(importances)
feature_names = list(X.columns)
plt.figure(figsize = (9, 9))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color = 'violet', align = 'center')
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.

Is Pruning for this Random Forest mode needed?

Pruning is already being applied through the hyperparameter tuning process:

- The max_depth parameter (6 to 7) limits how deep each tree can grow and so is a way form of pruning.
- The min_samples_leaf parameter (20 to 25) prevents splits that would create leaves with fewer samples than this threshold, effectively pruning potential branches.
- The max_features parameter (values 0.8 and 0.9) limits the number of features considered at each split, which indirectly controls tree complexity.
- The GridSearchCV finds the optimal parameters combination to balance model complexity and performance, which is essentially what pruning aims to achieve.

Therefore, additional pruning is not need.

Comparing the results between Decision Tree and Random Forest:

1. Overall Performance:

- Random Forest shows better overall accuracy (83% vs 80%)
- Random Forest has better weighted average metrics across the board

2. Class Balance:

- Decision Tree has slightly better recall for conversions (86% vs 85%)
- Random Forest has better precision for conversions (68% vs 62%)
- Random Forest has significantly better recall for non-conversions (83% vs 77%)

3. Error Patterns:

- Decision Tree produces more false positives (224 vs 167)
- Random Forest has slightly more false negatives (62 vs 57)
- Random Forest shows more balanced error distribution

4. Business Implications:

- Decision Tree identifies slightly more actual conversions (5 more)
- Random Forest has fewer false alarms (57 fewer)
- Random Forest correctly identifies more non-conversions (57 more)

Optimized Random Forest is the better choice because:

- Higher Overall Accuracy: 83% vs 80%
- Better Balanced Performance: More consistent across both classes
- Fewer False Positives: 167 vs 224, which means fewer wasted resources on non-converting leads
- Better Precision-Recall Balance: While the Decision Tree has slightly higher recall for conversions, the Random Forest's better precision means marketing efforts would be more efficiently targeted
- More Robust Predictions: Random Forests are generally less prone to overfitting than Decision Trees

Conclusion

Based on all the analysis above, the following conclusions can be drawn.

Insights:

Leads more likely to be converted to paid customers have following characteristics:

Demographic

- Professionals dominate both in numbers and conversion rate, followed by unemployed and students.
- Both professional and unemployed group have similar age range (35-60) and average age (50), while students are around 18 to 22.

Engagement Patterns:

- Website visit of about 2.5 - 5 times (assume every week)
- 2-5 pages view per visit
- Total time spent on website over 2 times of not converted
- High or medium level of profile completed dominates in numbers and conversion rate
- Trend to have intense emails communication near decision making stage, and have frequent websit vists before make decisions. This indicates that leads decide to enroll for their courses on websites after emails communication.

Marketing Channel:

- All of the channels such as digital channel, print media channels, educational channel are not effective in leads conversion.
- Low referral rate indicate that ExtraaLean's popularity is not enough.
- Website and mobile app are the most important touch point as first user interaction.

Conversion Factors:

- Time spent on website
- Website visits as the first interaction between leads and ExtraaLearn
- Profile completion level
- Current Occcupation
- Last activity type influences conversion probability
- Page views per visit
- Website visits

Typical profile of a lead which is likely to be converted to a paid customer:

- Employment status as professionals or unemployed
- Age around 42 to 58
- Have high or medium profile completed
- Have about 2-5 website visits
- Have 2-5 page view per website visit
- Have email activities as the last activity

Business Recommendations:

User Group marketing:

- Develop programs that can help professional to advance their career paths.
- Provide information about how the programs are related to career advancement.
- Implement occupation-specific marketing campaigns to attract unemployed leads.
- Provde discount packages or entry-level programs for students.

User Experience:

- Improve website navigation to attract 5+ page views.
- Improve personalized user journeys, for examples:
  - automatic recommendations of programs specific to leads profiles
  - auto email notice of new programs related to leads profiles
  - visualized learning paths for specific certifications and related programs
- As some leads decide not to join any course even they have many website visits. ExtraaLearn's website layout should provide better and efficient online users experiences such as
  - quick search by key words
  - system can go back to previous page when users visit again
  - clear display of important information such as course fee, course duration, certifications, etc

User Engagement:

- Strength website content to provide course information of hot subjects (AI, machine learning, etc), online chat bot to answer users queries, feedback from course graduates.
- Provide free trial of a small part of programs in order to promote more website visits.
- Keep close contact with leads by phone communication (text messaging should be used before phone call).
- Provide free-interest installment payments or discounts for unemployed leads or students.

Marketing Approach:

- Review and investigate the low usage and conversion rates of digital media and education channel.
- Develop more useful content for digital media and education channel.
- Provide course fee discount to attract referral opportunities.
- Cooperate with internationally famous colleges to provide hit programs, or launch promotion campaigns and career fairs. This strategy can enhance ExtraaLearn's popularity.

Course Content:

- Develop career related programs for leads (assume profession data will be collected).
- Provide diversified range of programs to attract professionals from different sectors.
- Develop courses of different durations to satisfy different needs.

Technical Improvements:

- Invest more in IT resources to maintain consistent performance of website and mobile app because they are both critical users interaction points.
- Website or mobile app response speed should be optimized to order to attract more visits or page views.