LEAD SCORING PROJECT

** lead scoring from business understanding to machine learning with pandas and scikit-learn or other required laibraries!

Problem context

Lead scoring is a process of assigning scores to prospects based on their profile and behavioral data in order to prioritize leads, improve close rates, and decrease buying cycles.

An education company named X Education sells online courses to industry professionals. On any given day, many professionals who are interested in the courses land on their website and browse for courses.

The company markets its courses on several websites and search engines like Google. Once these people land on the website, they might browse the courses or fill up a form for the course or watch some videos. When these people fill up a form providing their email address or phone number, they are classified to be a lead. Moreover, the company also gets leads through past referrals. Once these leads are acquired, employees from the sales team start making calls, writing emails, etc. Through this process, some of the leads get converted while most do not.

Business Goal¶

Goal from a business perspective: X Education wants to select the most promising leads, i.e. the leads that are most likely to convert into paying customers. The company requires you to build a model wherein you need to assign a lead score to each of the leads such that the customers with higher lead score have a higher conversion chance and the customers with lower lead score have a lower conversion chance. The CEO, in particular, has given a ballpark of the target lead conversion rate to be around 80%.

Goal from a Data Scientist perspective:

Our mission is to build a better lead scoring model, targeting an 80% conversion rate and precision score. Using predict_proba(), we'll assess lead probabilities. This project aims to gain insights and emphasize a data-driven approach for success.

** Prepare Work Environment**

Import Required Laibraries

In [10]:
```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import warnings
from scipy.stats import linregress, uniform
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKF
from sklearn.compose import make_column_transformer, make_column_selector
from sklearn.impute import KNNImputer, SimpleImputer
from sklearn.preprocessing import FunctionTransformer, OneHotEncoder, StandardScale
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import f1_score, recall_score, roc_auc_score, precision_score,
import os
os.getcwd()
```

Out[10]: 'C:\\Users\\aman'

In [11]:

```
## suppress warnings & display option
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', 50)
pd.set_option('display.max_rows', 50)
```

1. Load and Inspect the data

In [12]:

```
df = pd.read_csv("D:\\aman_new\\Lead Scoring Assignment\\Leads.csv",encoding = 'lat
df.head()
```

Out[12]:

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Tot Tim Spe o Websi
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.0	
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.0	67
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.0	153
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	No	0	1.0	30
4	3256f628-e534-4826-9d63-4a8b88782852	660681	Landing Page Submission	Google	No	No	1	2.0	142

1.1* shape and info about the dataset

In [ ]:

In [13]:
```
df.shape
```

Out[13]:
```
(9240, 37)
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 37 columns):
 #   Column                                         Non-Null Count  Dtype  
---  ------                                         --------------  -----  
 0   Prospect ID                                    9240 non-null   object 
 1   Lead Number                                    9240 non-null   int64  
 2   Lead Origin                                    9240 non-null   object 
 3   Lead Source                                    9204 non-null   object 
 4   Do Not Email                                   9240 non-null   object 
 5   Do Not Call                                    9240 non-null   object 
 6   Converted                                      9240 non-null   int64  
 7   TotalVisits                                    9103 non-null   float64
 8   Total Time Spent on Website                    9240 non-null   int64  
 9   Page Views Per Visit                           9103 non-null   float64
 10  Last Activity                                  9137 non-null   object 
 11  Country                                        6779 non-null   object 
 12  Specialization                                 7802 non-null   object 
 13  How did you hear about X Education             7033 non-null   object 
 14  What is your current occupation                6550 non-null   object 
 15  What matters most to you in choosing a course  6531 non-null   object 
 16  Search                                         9240 non-null   object 
 17  Magazine                                       9240 non-null   object 
 18  Newspaper Article                              9240 non-null   object 
 19  X Education Forums                             9240 non-null   object 
 20  Newspaper                                      9240 non-null   object 
 21  Digital Advertisement                          9240 non-null   object 
 22  Through Recommendations                        9240 non-null   object 
 23  Receive More Updates About Our Courses         9240 non-null   object 
 24  Tags                                           5887 non-null   object 
 25  Lead Quality                                   4473 non-null   object 
 26  Update me on Supply Chain Content              9240 non-null   object 
 27  Get updates on DM Content                      9240 non-null   object 
 28  Lead Profile                                   6531 non-null   object 
 29  City                                           7820 non-null   object 
 30  Asymmetrique Activity Index                    5022 non-null   object 
 31  Asymmetrique Profile Index                     5022 non-null   object 
 32  Asymmetrique Activity Score                    5022 non-null   float64
 33  Asymmetrique Profile Score                     5022 non-null   float64
 34  I agree to pay the amount through cheque       9240 non-null   object 
 35  A free copy of Mastering The Interview         9240 non-null   object 
 36  Last Notable Activity                          9240 non-null   object 
dtypes: float64(4), int64(3), object(30)
memory usage: 2.6+ MB
```

- Initial thoughts and action plan
  - check duplicates
  - drop prospect Id and lead number.
  - reformat column names without space and lowercase for practicality, and change some columns name for others names more intuitive.
  - seems that we got lot of binary columns, with only yes or no values etc. Convert it in binary encoding.

- decide what to do all the "select" data, if count it as null or assign another category like "not answered" values.

1.2 check if columns specified are really binary

In [15]:
```
binary_cats = ['Do Not Email','Do Not Call','Search','Magazine','Newspaper Article'
               'X Education Forums','Newspaper','Digital Advertisement','Through Re
               'Receive More Updates About Our Courses', 'Update me on Supply Chain
               'I agree to pay the amount through cheque', 'A free copy of Masterin

null_values = df[binary_cats].isnull().sum()
total = df[binary_cats].count()
yes_no = df[binary_cats].applymap(lambda x: 1 if x == 'Yes' or x == 'No' else 0).su
df_binary_cats = pd.DataFrame({'total': total,
                               'null_%': null_values/total*100,
                               'yes/no_%': yes_no/total*100})
df_binary_cats
```

Out[15]:

	total	null_%	yes/no_%
Do Not Email	9240	0.0	100.0
Do Not Call	9240	0.0	100.0
Search	9240	0.0	100.0
Magazine	9240	0.0	100.0
Newspaper Article	9240	0.0	100.0
X Education Forums	9240	0.0	100.0
Newspaper	9240	0.0	100.0
Digital Advertisement	9240	0.0	100.0
Through Recommendations	9240	0.0	100.0
Receive More Updates About Our Courses	9240	0.0	100.0
Update me on Supply Chain Content	9240	0.0	100.0
Get updates on DM Content	9240	0.0	100.0
I agree to pay the amount through cheque	9240	0.0	100.0
A free copy of Mastering The Interview	9240	0.0	100.0

- there is no missing values
- All the columns have only "yes" or "no" values.

1.3. Separate train and test datasets

Let's separate train and test set before keep seeing more info. Separating train and test data is essential to avoid data leakage, evaluate model generalization, and make unbiased performance assessments in machine learning. It ensures robust model development and reliable predictions on new, unseen data.

Why stratify by target label? Stratifying train and test datasets in classification ensures balanced class representation, guarding against biased or imbalanced model learning. It promotes accurate evaluation, preventing skewed performance metrics.

In [16]:
```
train, test = train_test_split(df, test_size=.2, random_state=12, stratify=df['Conv
print(f'train shape: {train.shape}')
print(f'test shape: {test.shape}')
```

```
train shape: (7392, 37)
test shape: (1848, 37)
```

1.4. Inspecting only training dataset

In [17]:
```
print(f'In the train set are {train.duplicated().sum()} duplicates')
```

```
In the train set are 0 duplicates
```

check the values in Asymmetrique profile index columns

In [18]:
```
train['Asymmetrique Profile Index'].value_counts(dropna=False)
```

Out[18]:
```
Asymmetrique Profile Index
NaN          3362
02.Medium    2243
01.High      1762
03.Low         25
Name: count, dtype: int64
```

In [19]:
```
train['Asymmetrique Activity Index'].value_counts(dropna=False)
```

Out[19]:
```
Asymmetrique Activity Index
NaN          3362
02.Medium    3080
01.High       648
03.Low        302
Name: count, dtype: int64
```

Assymetrique's columns treatment:

We have identified three distinct categories and some missing records. To improve the data's representativenes for machine learning modeling we will focus on the integer values and reverse their order, emphasizing a higher-is-better perspective.

-

High: Assigned a value of * 3 Medium: Assigned a value o *f 2 Low: Assigned a value of 1

2. Data cleaning & Feature Engineering

2.1 Data Cleaning

Let us embark on our first data cleaning endeavor! Our strategy involves transforming each step into Scikit-learn transformation objects, harmonizing the entire process into a unified pipeline.

In [20]:
```
def data_cleaning(df):
  """Do some of the data cleaning procedures that we
  specified at the begining of the notebook"""
  # drop columns id columns
  df = df.drop(['Prospect ID','Lead Number'], axis=1)

  # asymmetrique index columns transformation
  df['Asymmetrique Activity Index'] = df['Asymmetrique Activity Index'].str.split('
                                                                       .str.replac
                                                                       .str.replac
                                                                       .astype(np.
                                                                              )
  df['Asymmetrique Profile Index'] = df['Asymmetrique Profile Index'].str.split('.'
                                                                       .str.replac
                                                                       .str.replac
                                                                       .astype(np.
                                                                              )
  # binary encoding
  df[binary_cats] = df[binary_cats].applymap(lambda x: 0 if x == 'No' else 1)

  # rename columns for practicity
  df.columns = df.columns.str.replace(' ','_').str.lower()
  return df

# Convert custom function into transformer
initial_clean = FunctionTransformer(data_cleaning)

train_clean = initial_clean.fit_transform(train);
```

4.2 Inspecting category columns

In this stage, we'll first inspect the categorical columns from a practical and business-oriented perspective, before delving into more advanced statistical analysis.

I firmly believe that simplicity often holds the key to effective solutions.

The goal is to take a first look through all category columns to do some feature engineering, extract some initial thoughts for future EDA/feature engineer and handling missing values.

In [21]:
```
train_clean.lead_origin.value_counts(dropna=False)
```

Out[21]:
```
lead_origin
Landing Page Submission    3906
API                        2889
Lead Add Form               550
Lead Import                  47
Name: count, dtype: int64
```

In [22]:
```
train_clean.lead_source.value_counts(dropna=False)
```

Out[22]:
```
lead_source
Google               2326
Direct Traffic       2033
Olark Chat           1408
Organic Search        916
Reference             405
Welingak Website      111
Referral Sites         98
Facebook               46
NaN                    27
bing                    6
Click2call              4
google                  3
Live Chat               2
blog                    1
testone                 1
Social Media            1
youtubechannel          1
WeLearn                 1
Press_Release           1
Pay per Click Ads       1
Name: count, dtype: int64
```

In [23]:
```
train_clean.last_activity.value_counts(dropna=False)
```

Out[23]:
```
last_activity
Email Opened                    2712
SMS Sent                        2224
Olark Chat Conversation          793
Page Visited on Website          506
Converted to Lead                336
Email Bounced                    271
Email Link Clicked               211
Form Submitted on Website         97
NaN                               80
Unreachable                       71
Unsubscribed                      50
Had a Phone Conversation          23
Approached upfront                 8
View in browser link Clicked       5
Email Received                     2
Resubscribed to emails             1
Email Marked Spam                  1
Visited Booth in Tradeshow         1
Name: count, dtype: int64
```

In [24]:
```
train_clean.country.value_counts(dropna=False)
```

Out[24]:
```
country
India                   5201
NaN                     1954
United States             57
United Arab Emirates      49
Singapore                 21
Saudi Arabia              17
United Kingdom            13
Australia                  9
Qatar                      9
Bahrain                    6
Oman                       4
Nigeria                    4
Germany                    4
France                     4
unknown                    4
Kuwait                     3
Hong Kong                  3
Canada                     3
South Africa               3
China                      2
Belgium                    2
Sweden                     2
Italy                      2
Bangladesh                 2
Ghana                      2
Netherlands                2
Switzerland                1
Denmark                    1
Uganda                     1
Tanzania                   1
Russia                     1
Philippines                1
Malaysia                   1
Sri Lanka                  1
Asia/Pacific Region        1
Kenya                      1
Name: count, dtype: int64
```

In [25]:
```
train_clean.specialization.value_counts(dropna=False)
```

Out[25]:
```
specialization
Select                               1554
NaN                                  1154
Finance Management                    780
Marketing Management                  682
Human Resource Management             661
Operations Management                 407
Business Administration               329
IT Projects Management                295
Supply Chain Management               281
Banking, Investment And Insurance     272
Media and Advertising                 161
Travel and Tourism                    155
International Business                152
Healthcare Management                 128
Hospitality Management                 89
E-COMMERCE                             84
Retail Management                      76
Rural and Agribusiness                 59
E-Business                             43
Services Excellence                    30
Name: count, dtype: int64
```

In [26]:
```
train_clean.how_did_you_hear_about_x_education.value_counts(dropna=False)
```

Out[26]:
```
how_did_you_hear_about_x_education
Select                     4023
NaN                        1768
Online Search               647
Word Of Mouth               288
Student of SomeSchool       258
Other                       139
Multiple Sources            128
Social Media                 51
Advertisements               49
SMS                          21
Email                        20
Name: count, dtype: int64
```

In [27]:
```
train_clean.what_is_your_current_occupation.value_counts(dropna=False)
```

Out[27]:
```
what_is_your_current_occupation
Unemployed                4474
NaN                       2159
Working Professional       559
Student                    172
Other                       14
Housewife                    8
Businessman                  6
Name: count, dtype: int64
```

In [28]:
```
train_clean.what_matters_most_to_you_in_choosing_a_course.value_counts(dropna=False
```

Out[28]:
```
what_matters_most_to_you_in_choosing_a_course
Better Career Prospects      5216
NaN                          2173
Flexibility & Convenience       2
Other                           1
Name: count, dtype: int64
```

In [29]:
```
train_clean.tags.value_counts(dropna=False)
```

Out[29]:
```
tags
NaN                                                  2687
Will revert after reading the email                  1664
Ringing                                               958
Interested in other courses                           418
Already a student                                     365
Closed by Horizzon                                    287
switched off                                          200
Busy                                                  146
Lost to EINS                                          138
Not doing further education                           110
Interested  in full time MBA                           92
Graduation in progress                                 87
invalid number                                         60
Diploma holder (Not Eligible)                          51
wrong number given                                     42
opp hangup                                             29
number not provided                                    21
in touch with EINS                                      7
Want to take admission but has financial problems       6
Lost to Others                                          5
Still Thinking                                          4
In confusion whether part time or DLP                   4
Interested in Next batch                                3
Lateral student                                         3
University not recognized                               2
Shall take in the next coming month                     2
Recognition issue (DEC approval)                        1
Name: count, dtype: int64
```

In [30]:
```
train_clean.lead_quality.value_counts(dropna=False)
```

Out[30]:
```
lead_quality
NaN                  3796
Might be             1249
Not Sure              884
High in Relevance     508
Worst                 490
Low in Relevance      465
Name: count, dtype: int64
```

In [31]:
```
train_clean.city.value_counts(dropna=False)
```

Out[31]:
```
city
Mumbai                         2581
Select                         1806
NaN                            1139
Thane & Outskirts               607
Other Cities                    539
Other Cities of Maharashtra     349
Other Metro Cities              307
Tier II Cities                   64
Name: count, dtype: int64
```

In [32]:
```python
train_clean.last_notable_activity.value_counts(dropna=False)
```

Out[32]:
```
last_notable_activity
Modified                        2734
Email Opened                    2219
SMS Sent                        1759
Page Visited on Website          259
Olark Chat Conversation          153
Email Link Clicked               140
Email Bounced                     49
Unsubscribed                      39
Unreachable                       24
Had a Phone Conversation          10
Form Submitted on Website          1
Email Received                     1
View in browser link Clicked       1
Resubscribed to emails             1
Email Marked Spam                  1
Approached upfront                 1
Name: count, dtype: int64
```

Initial feature Engineering

Apply initial changes described in the previous insights

In [33]:
```python
def initial_feature_engineering(df):
  """Do some feature engineering"""
  # lead_source
  df['lead_source'] = df['lead_source'].str.replace('|'.join(['google','Pay per Cli
  df['lead_source'] = df['lead_source'].apply(lambda x: "Referral Sites" if 'blog'
  df['lead_source'] = df['lead_source'].str.replace('Live Chat','Olark Chat')
  df['lead_source'] = df['lead_source'].str.replace('bing','Organic Search')
  df['lead_source'] = df[df['lead_source'] != 'Other'].lead_source.apply(lambda x:
  # last_activity and last_notable_activity
  activity = ['last_activity','last_notable_activity']
  df[activity] = df[activity].apply(lambda x: x.str.replace('|'.join(['Email Receiv
  df[activity] = df[activity].apply(lambda x: x.str.replace('|'.join(['Email Marked
  df[activity] = df[activity].apply(lambda x: x.str.replace('Resubscribed to emails
  df[activity] = df[activity].apply(lambda x: x.str.replace('|'.join(['Visited Boot
  # country
  df['country'] = df['country'].apply(lambda x: np.nan if x in ['Unknown','unknown'
  # specialization
  df['specialization'] = df['specialization'].str.replace('|'.join(['E-COMMERCE','E
```

```
    df['specialization'] = df['specialization'].str.replace('Banking, Investment And
    df['specialization'] = df['specialization'].str.replace('Media and Advertising','
    df['specialization'] = df['specialization'].str.replace('Select','Not Provided')
    # how_did_you_hear
    df['how_did_you_hear_about_x_education'] = df['how_did_you_hear_about_x_education
    df['how_did_you_hear_about_x_education'] = df['how_did_you_hear_about_x_education
    # importance_in_course
    df['what_matters_most_to_you_in_choosing_a_course'] = df['what_matters_most_to_yo
    # lead_profile
    df['lead_profile'] = df['lead_profile'].str.replace('Select','Not Assigned')
    # city
    df['city'] = df['city'].str.replace('Select','Not Provided')

    return df

initial_feature_engineering = FunctionTransformer(initial_feature_engineering)
train_clean = initial_feature_engineering.fit_transform(train_clean)
```

3. Explore Missing values

copy of the dataset and visualizations style

In [34]:
```
train_ = train_clean.copy()

# Set style for better visualizations
train_eda = train.copy()
sns.set_style('dark')
sns.set(rc={'axes.grid':False})
sns.set_palette('viridis')
```

In [35]:
```
null_ = pd.DataFrame()
null_['proportion'] = np.round(train_clean.isnull().sum()/len(train_clean),4) * 100
null_['amount'] = train_clean.isnull().sum()

# Show only those columns with at least 1 missing value
null_.sort_values(by='proportion', ascending=False)[null_.amount > 0]
```

Out[35]:

	proportion	amount
lead_quality	51.35	3796
asymmetrique_activity_index	45.48	3362
asymmetrique_profile_score	45.48	3362
asymmetrique_profile_index	45.48	3362
asymmetrique_activity_score	45.48	3362
tags	36.35	2687
lead_profile	29.40	2173
what_matters_most_to_you_in_choosing_a_course	29.40	2173
what_is_your_current_occupation	29.21	2159
country	26.50	1959
how_did_you_hear_about_x_education	23.92	1768
specialization	15.61	1154
city	15.41	1139
page_views_per_visit	1.45	107
totalvisits	1.45	107
last_activity	1.08	80

** insights * missing values in certain columns, often requiring employee input, might stem from uncategorized leads. streamlining lead management can improve data collection, inform decision-making, and optimize lead conversion strategies. Further investigation is necessary to confirm this hypothesis

define plot functions

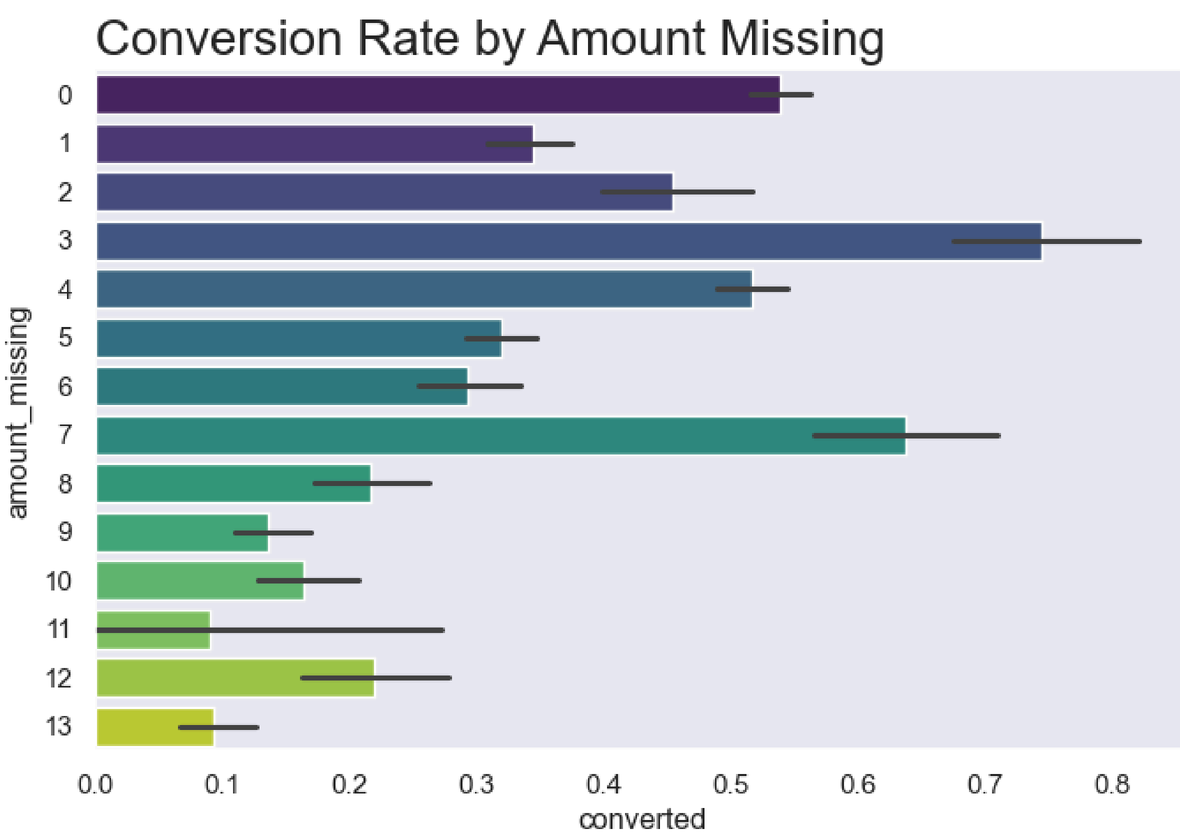
In [36]:

```
def barplot_catcols(column,width,heigh):
  """Plot conversion rate"""
  fig, ax  = plt.subplots(figsize=(width,heigh))
  ax = sns.barplot(data=train_.fillna('NaN'), x='converted', y=column,
            order=order(train_.fillna('NaN'),column),
            orient='h', palette='viridis',
            seed=2)
  plt.title(f'Conversion Rate by {column.replace("_"," ").title()}', loc='left', si
  return ax

def order(df,x,y=None):
    if y is not None:
        return df.groupby(x)[y].mean().sort_values(ascending=False).index
```

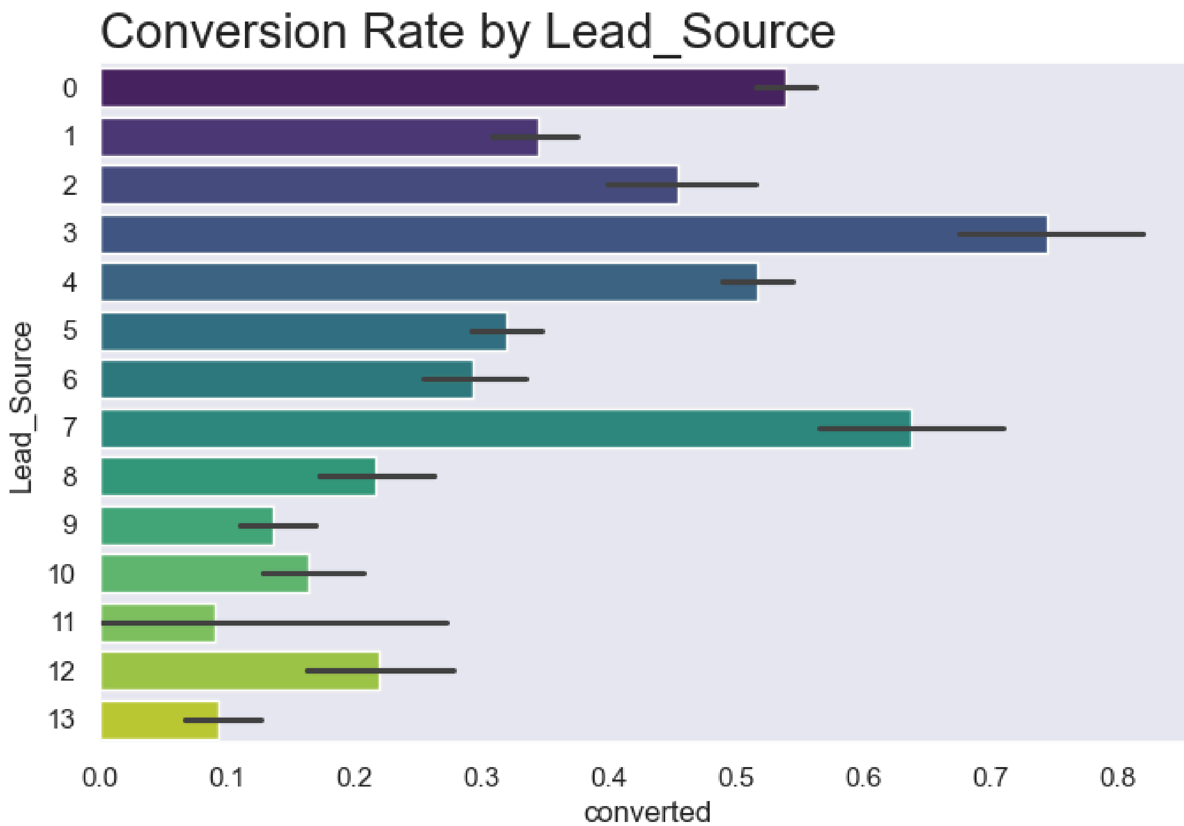
```
    else:
        return df.groupby(x)['converted'].mean().sort_values(ascending=False).index
```

How much of the missing values belong to the same people?

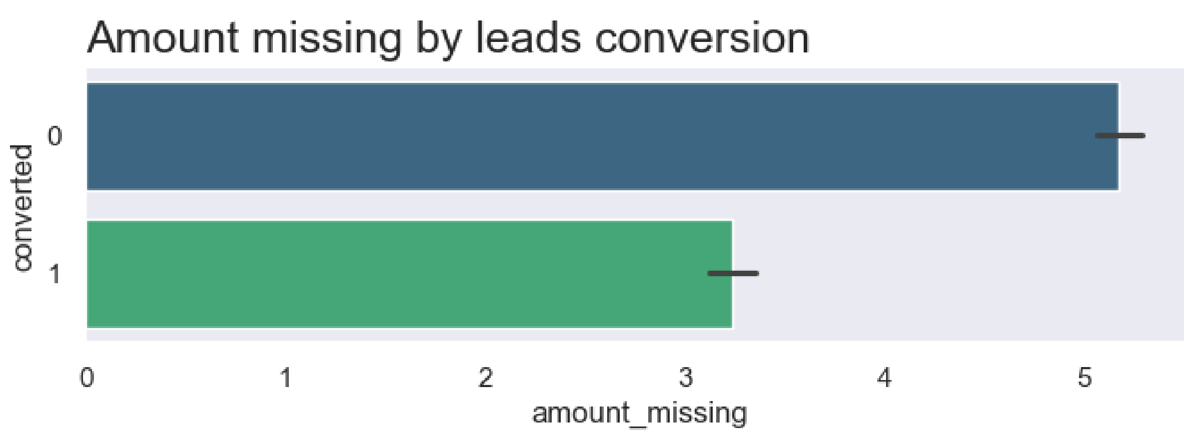
In [37]:
```
# Number of missing values in each row
train_['amount_missing'] = train_.isnull().sum(1)

# Plot the relation between amount missing and conversion rate
fig, ax  = plt.subplots(figsize=(8,5))
ax = sns.barplot(data=train_.fillna('NaN'), x='converted', y='amount_missing',
            orient='h', palette='viridis',
            seed=2)
plt.title(f'Conversion Rate by Amount Missing', loc='left', size=20)
plt.show()
```



In [38]:
```
# Number of missing values in each row
train_['Lead_Source'] = train_.isnull().sum(1)

# Plot the relation between amount missing and conversion rate
fig, ax  = plt.subplots(figsize=(8,5))
ax = sns.barplot(data=train_.fillna('NaN'), x='converted', y='Lead_Source',
            orient='h', palette='viridis',
            seed=2)
plt.title(f'Conversion Rate by Lead_Source', loc='left', size=20)
plt.show()
```



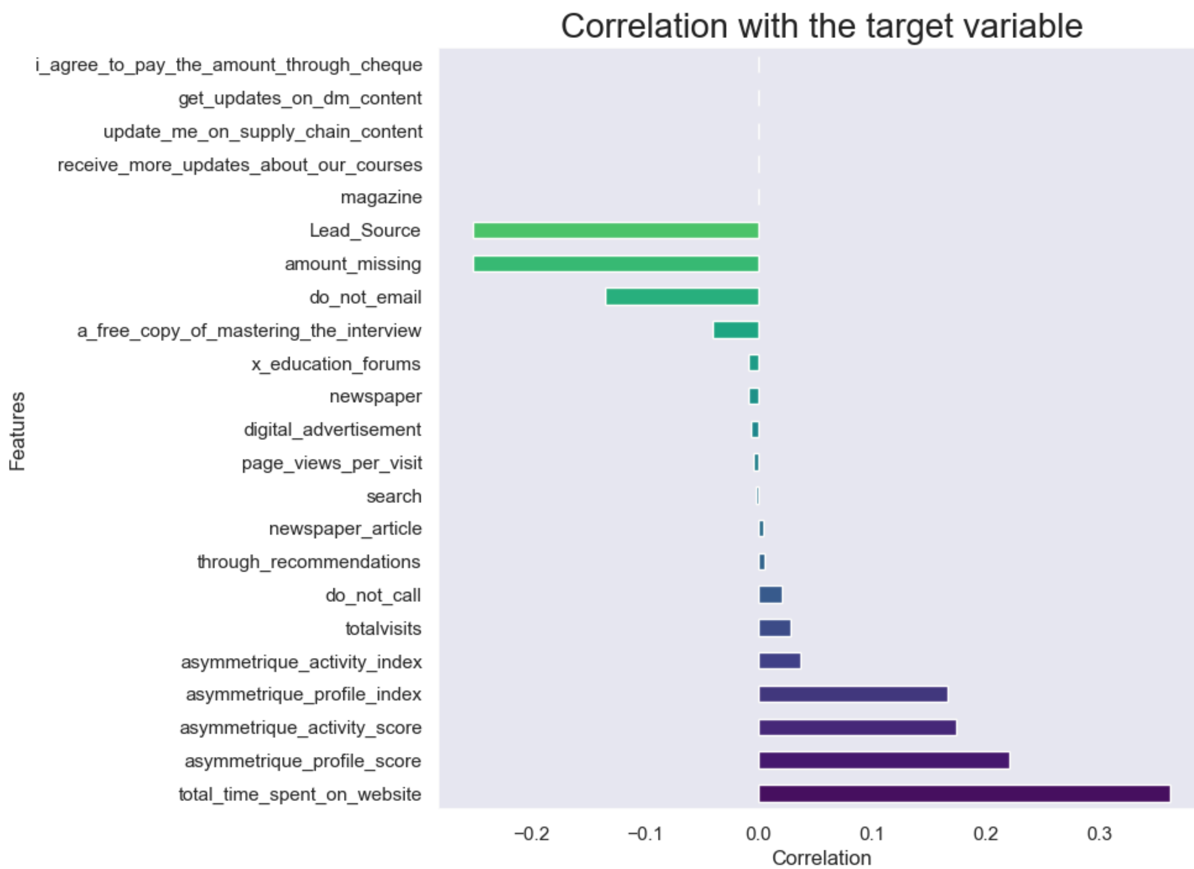
In [39]:
```
fig, ax  = plt.subplots(figsize=(8,2))
ax = sns.barplot(data=train_, x='amount_missing', y='converted',
             orient='h', palette=sns.color_palette('viridis',2),
             seed=2)
plt.title(f'Amount missing by leads conversion', loc='left', size=18)
plt.show()
```



3.2 Correlation of numerical columns with converted column¶

In [40]:
```
correlations = train_.select_dtypes('number').corr()['converted'].sort_values(ascen

plt.figure(figsize=(8, 8))
correlations[1:].plot(kind='barh',
```

```
                    color=sns.color_palette('viridis', len(correlations)))

plt.title('Correlation with the target variable', fontsize=20)
plt.xlabel('Correlation')
plt.ylabel('Features')
plt.show()
```



In [41]:
```
print(f'Duplicate rows from original dataset: {train.duplicated().sum()}')
print(f'Duplicate rows after feature engineer: {train_clean.duplicated().sum()}')
```

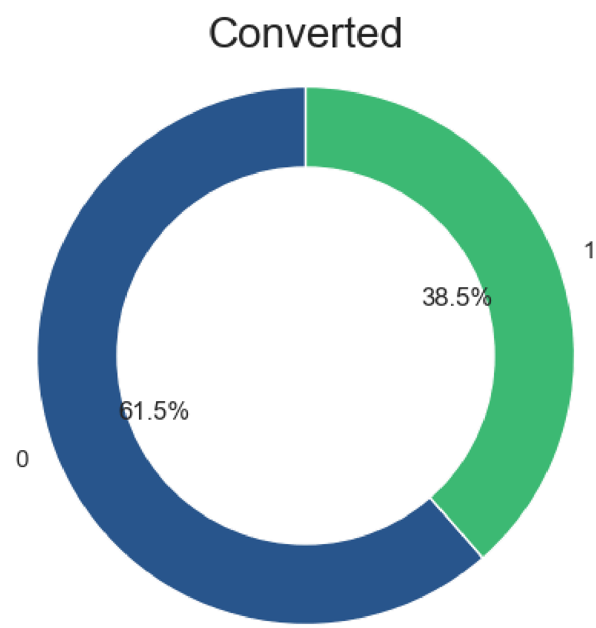
```
Duplicate rows from original dataset: 0
Duplicate rows after feature engineer: 984
```

Handle Missing Values: We currently lack sufficient information to determine the best approach for dealing with missing values. To address this, we will conduct a detailed data exploration, searching for patterns related to lead conversion. Once we have a clearer understanding, we can devise the most appropriate strategy for handling these missing records.

4. Exploratory Data Analysis

¶ Considering the prevalence of categorical or binary variables, we'll treat "NaN" values as a distinct category for comparison. For numerical columns with few "NaN" values, we'll exclude them to ensure robust analysis. This follows EDA best practices for gaining valuable insights from the dataset.

In [42]:
```
count = train_['converted'].value_counts()

fig, ax = plt.subplots(figsize=(10, 5))
ax.pie(count, labels=count.index, autopct='%1.1f%%', startangle=90, colors=['#29568
ax.set_title('Converted', size=20)

centre_circle = plt.Circle((0,0),0.70,fc='white')
fig.gca().add_artist(centre_circle)

plt.axis('equal')
plt.show()
```



Insight: The dataset exhibits a relatively balanced distribution of the target variable. While there may be some variations in class proportions, it's not extremely unbalanced.

In [43]:
```
train_.loc[:,'asymmetrique_activity_index':'asymmetrique_profile_score'].corr().sty
```

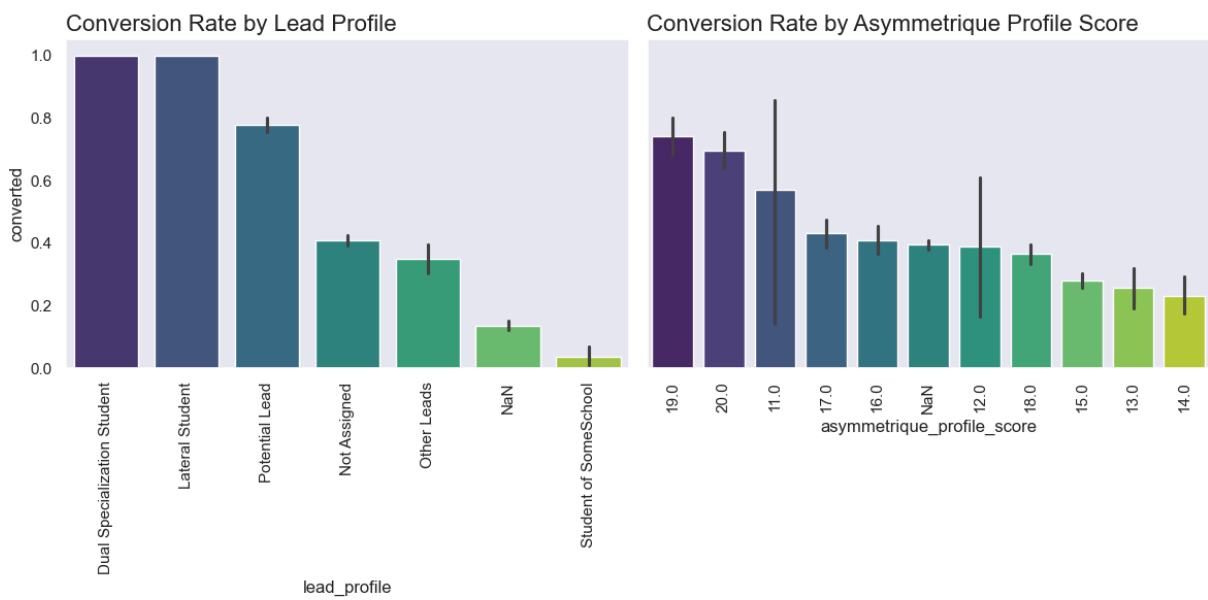
Out[43]:

	asymmetrique_activity_index	asymmetrique_profile_index	asyn
asymmetrique_activity_index	1.000000	-0.145399	
asymmetrique_profile_index	-0.145399	1.000000	
asymmetrique_activity_score	0.855985	-0.145366	
asymmetrique_profile_score	-0.122669	0.883177	

Insight: As expected, there's a strong correlation between the "Score" and "Index" columns. Given the level of detail in the data, retaining the score columns appears to be a sound choice. These columns appear to offer valuable information, and their inclusion in our analysis is likely to yield valuable insights.

4.1 . Categorical Variables

In [44]:
```
fig, ax  = plt.subplots(1,2, figsize=(12,6), sharey=True)

sns.barplot(data=train_.fillna('NaN'), x='lead_profile', y='converted',
            palette='viridis', order=order(train_.fillna('NaN'),'lead_profile'),
            seed=2, ax=ax[0])
ax[0].set_xticklabels(ax[0].get_xticklabels(), rotation=90)
ax[0].set_title(f'Conversion Rate by Lead Profile', loc='left', size=16)

sns.barplot(data=train_.fillna('NaN'), x='asymmetrique_profile_score', y='converted
                  palette='viridis', order=order(train_.fillna('NaN'),'asymmetrique
                    seed=2, ax=ax[1])
ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=90)
ax[1].set_title(f'Conversion Rate by Asymmetrique Profile Score', loc='left', size=

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



Insights: There's a significant difference in the conversion rate of people with "Not Assigned" and "NaN" values, which might suggest that they do not belong to the same group. Profile Score could be a better predictor than Lead Profile, as the conversion rate tends to increase with higher scores. Because both columns essentially represent the same information, it is advisable to drop the Lead Profile column for simplicity and clarity in the analysis.

4.2 Lead Activity

- Correlation between activity track record (columns related with the web) and activity/profile score

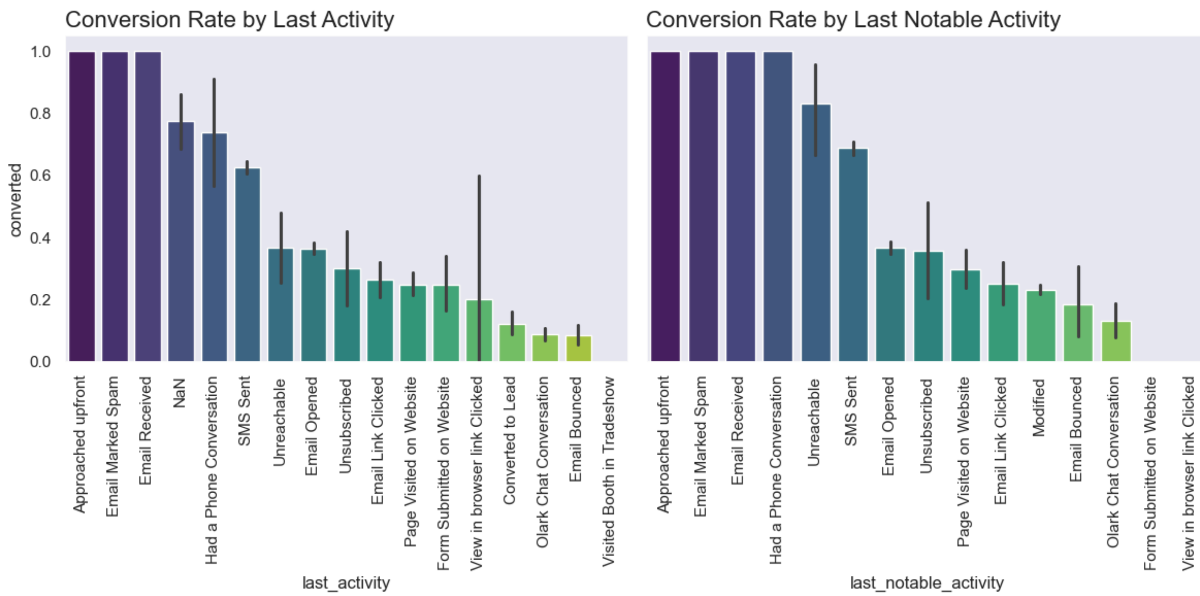
In [45]:
```
activity_columns = ['totalvisits','total_time_spent_on_website','page_views_per_vis
                    'asymmetrique_profile_score','asymmetrique_activity_score']

train_[activity_columns].corr().style.background_gradient(cmap='vlag_r')
```

Out[45]:

	totalvisits	total_time_spent_on_website	page_views_per_visit	a
totalvisits	1.000000	0.261952	0.598883	
total_time_spent_on_website	0.261952	1.000000	0.323684	
page_views_per_visit	0.598883	0.323684	1.000000	
asymmetrique_profile_score	0.129016	0.167992	0.165945	
asymmetrique_activity_score	-0.061397	-0.066008	-0.171264	

- Having columns about last activity and last notable activity provides more information?

In [46]:
```
fig, ax  = plt.subplots(1,2, figsize=(12,6), sharey=True)

sns.barplot(data=train_.fillna('NaN'), x='last_activity', y='converted',
            order=order(train_.fillna('NaN'),'last_activity'),
            palette='viridis',
            seed=2, ax=ax[0])
ax[0].set_xticklabels(ax[0].get_xticklabels(), rotation=90)
ax[0].set_title(f'Conversion Rate by Last Activity', loc='left', size=16)

sns.barplot(data=train_.fillna('NaN'), x='last_notable_activity', y='converted',
                  order=order(train_.fillna('NaN'),'last_notable_activity'),
                  palette='viridis', seed=2)
ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=90)
ax[1].set_title(f'Conversion Rate by Last Notable Activity', loc='left', size=16)

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



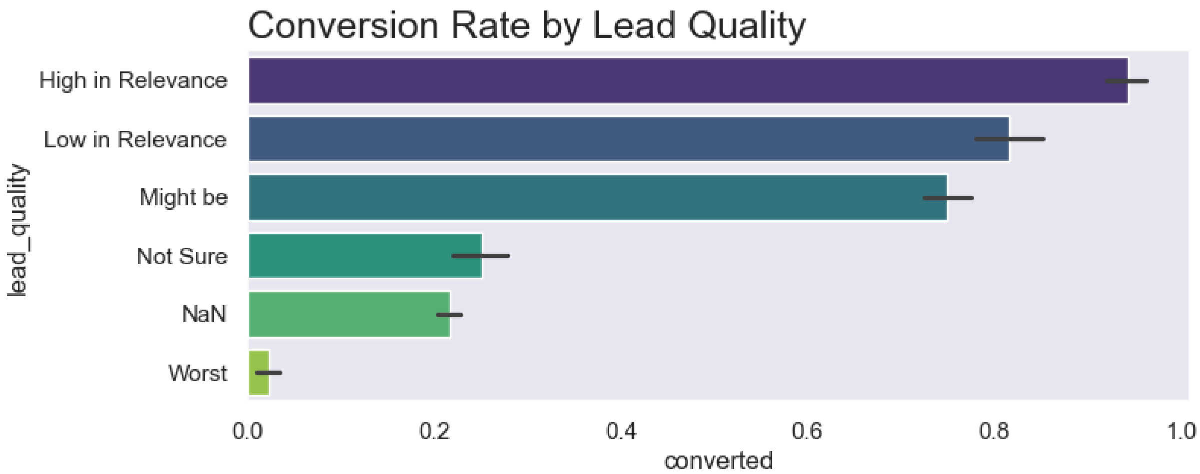
Insights: Activity Score seems to be a less effective predictor compared to Profile Score. No significant relationship was found between Activity Index and Last Activity, Last Notable Activity, or columns related to visits. There's no significant correlation among columns related to visits. Last Activity does not appear to provide substantially more information than Last Notable Activity. Hence, it may be preferable to retain Last Notable Activity.

** Business Suggestion:

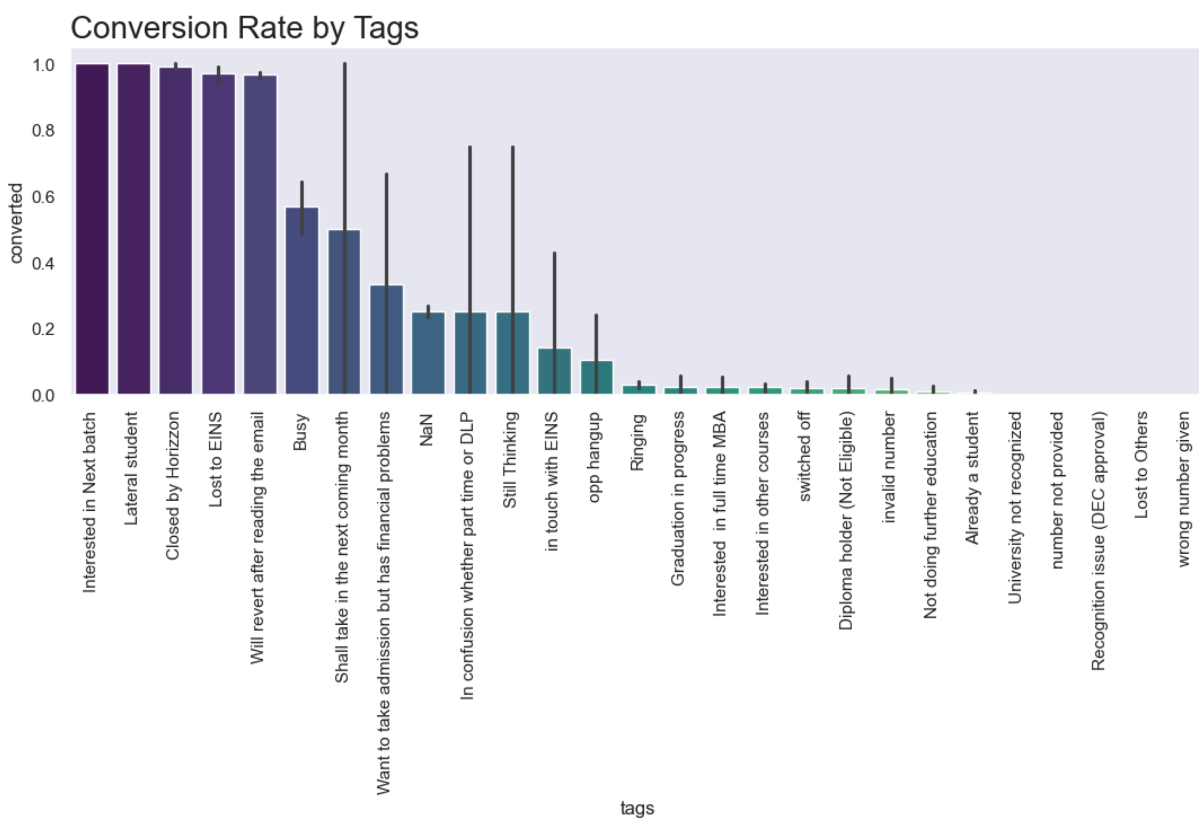
```
*Our analysis reveals a significant correlation between phone
conversations and lead conversions. To maximize results, consider
increasing phone calls to leads. Prioritizing "Hot Leads" for calls
can enhance resource allocation and boost conversion rates,
ultimately driving better business outcomes.
```

4.3 Lead Quality and Tags

In [47]:
```
barplot_catcols('lead_quality',8,3)
plt.show()
```



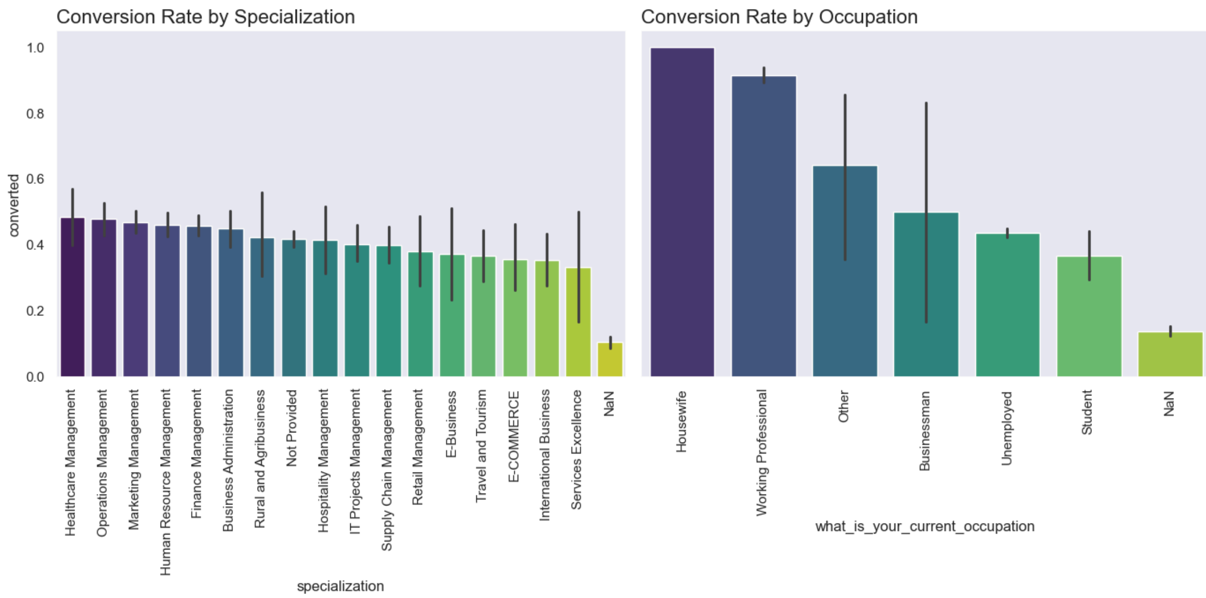
In [48]:
```
fig, ax  = plt.subplots(figsize=(13,4))

sns.barplot(data=train_.fillna('NaN'), x='tags', y='converted',
            order=order(train_.fillna('NaN'),'tags'),
            palette='viridis',
            seed=2)
plt.xticks(rotation=90)
plt.title(f'Conversion Rate by Tags', loc='left', size=20)
plt.show()
```



4.4 Ocupation and Specialization

In [49]:
```
fig, ax  = plt.subplots(1,2, figsize=(14,7), sharey=True)
```

```
sns.barplot(data=train_.fillna('NaN'), x='specialization', y='converted',
            order=order(train_.fillna('NaN'),'specialization'),
            palette='viridis',
            seed=2, ax=ax[0])
ax[0].set_xticklabels(ax[0].get_xticklabels(), rotation=90)
ax[0].set_title(f'Conversion Rate by Specialization', loc='left', size=16)

sns.barplot(data=train_.fillna('NaN'), x='what_is_your_current_occupation', y='conv
                  order=order(train_.fillna('NaN'),'what_is_your_current_occupation
                  palette='viridis', seed=2)
ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=90)
ax[1].set_title(f'Conversion Rate by Occupation', loc='left', size=16)

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



- Number of missing values for each row in these two categories

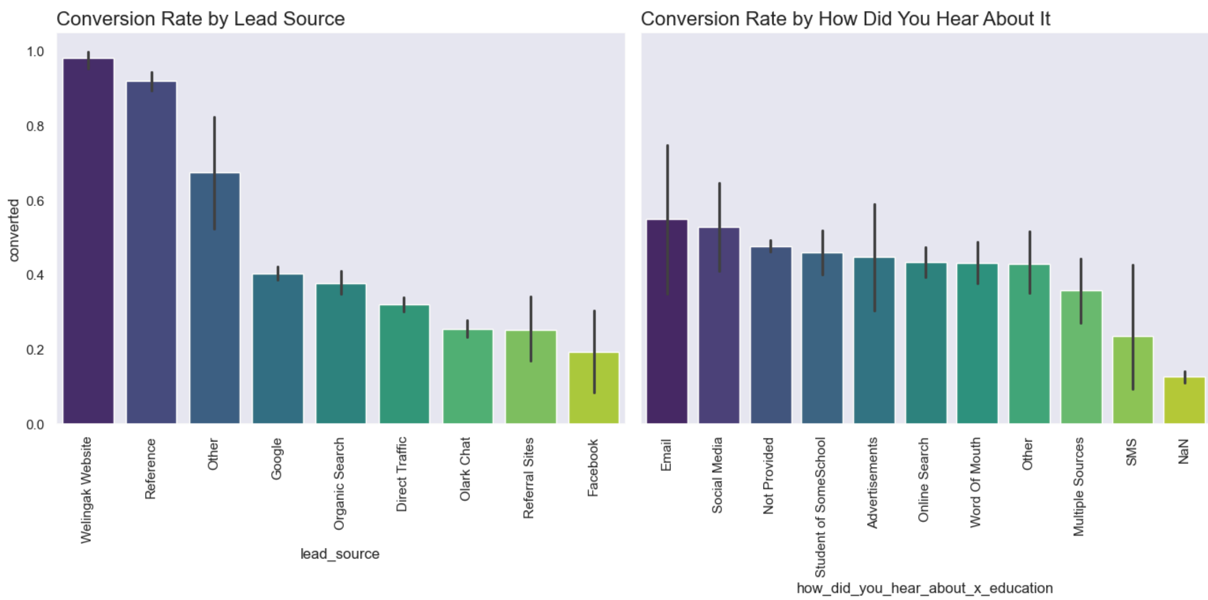
In [50]:
```
train_[['what_is_your_current_occupation','specialization']].isnull().sum(1).value_
```

Out[50]:
```
0    5220
2    1141
1    1031
Name: count, dtype: int64
```

** 4.5. The source from which the customer heard about X Education and the source of the lead**

In [51]:
```
fig, ax  = plt.subplots(1,2, figsize=(14,7), sharey=True)

sns.barplot(data=train_.fillna('NaN'), x='lead_source', y='converted',
            order=order(train_.fillna('NaN'),'lead_source'),
            palette='viridis',
            seed=2, ax=ax[0])
ax[0].set_xticklabels(ax[0].get_xticklabels(), rotation=90)
ax[0].set_title(f'Conversion Rate by Lead Source', loc='left', size=16)
```

```
sns.barplot(data=train_.fillna('NaN'), x='how_did_you_hear_about_x_education', y='c
                  order=order(train_.fillna('NaN'),'how_did_you_hear_about_x_educat
                  palette='viridis', seed=2)
ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=90)
ax[1].set_title(f'Conversion Rate by How Did You Hear About It', loc='left', size=1

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



Business Suggestion:

Referrals, with a 90% conversion rate, are a top-performing lead source due to their trustworthiness. To capitalize on this potential, the business sshould incentivize, personalize, track, showcase testimonials, and leverage word-of-mouth marketing for effective growth

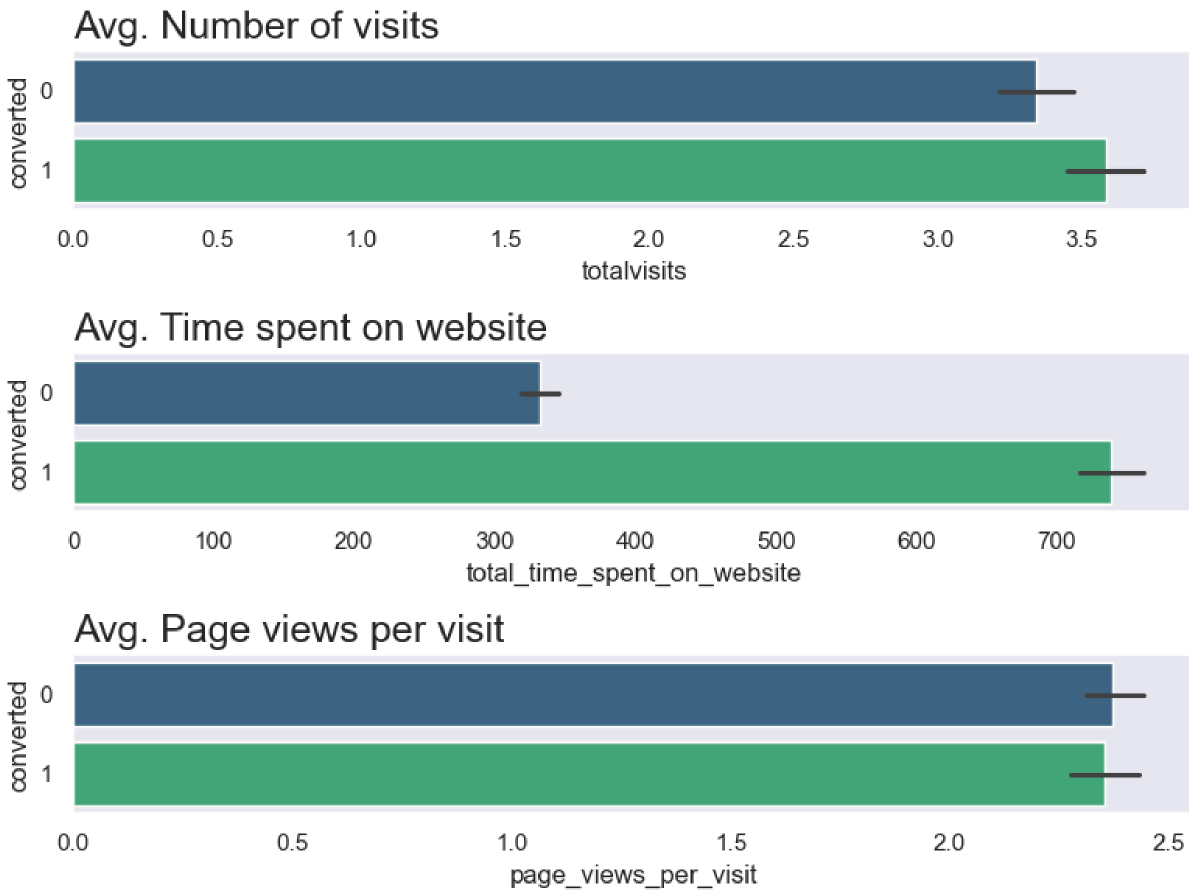
Numaric Variables`

In [52]:
```
train_.select_dtypes(include=['number']).nunique().sort_values()
```

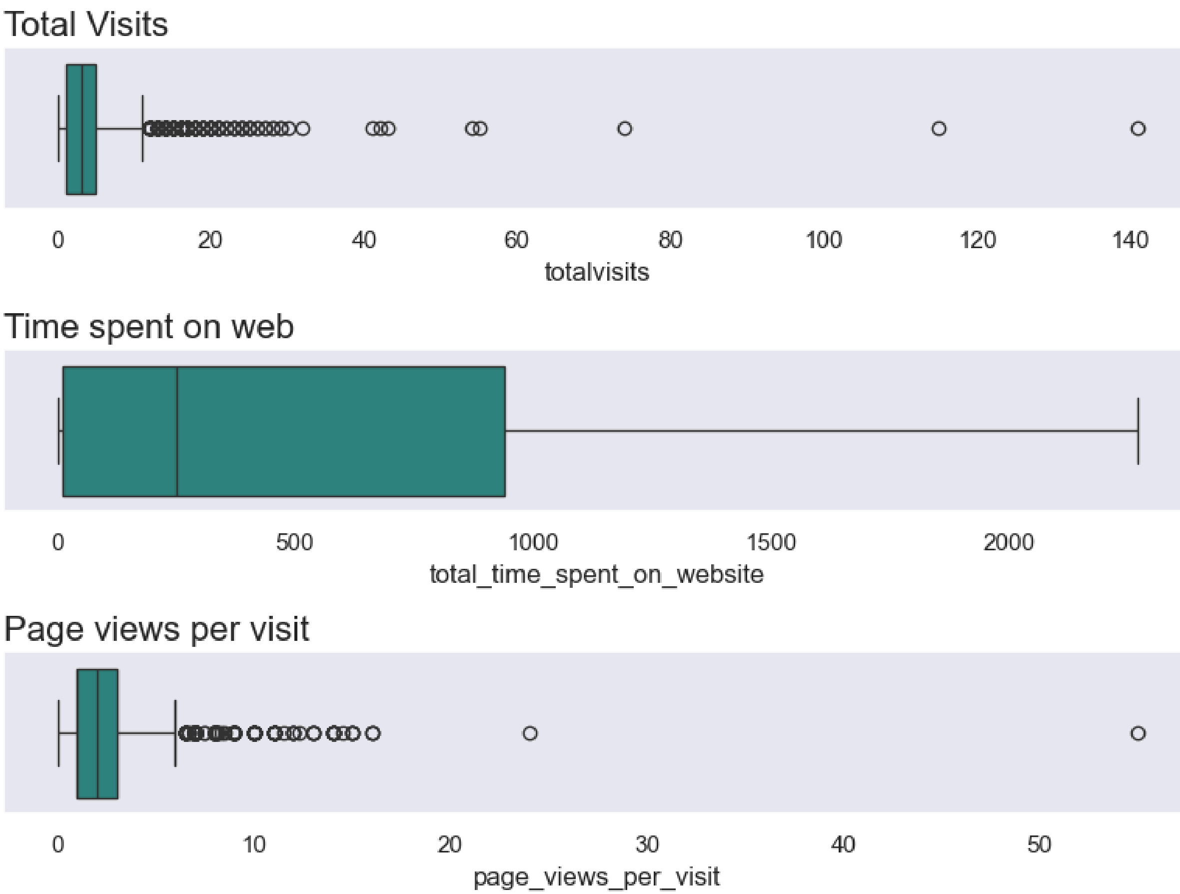
Out[52]:
```
i_agree_to_pay_the_amount_through_cheque       1
get_updates_on_dm_content                      1
update_me_on_supply_chain_content              1
receive_more_updates_about_our_courses         1
magazine                                       1
do_not_email                                   2
a_free_copy_of_mastering_the_interview         2
through_recommendations                        2
newspaper                                      2
digital_advertisement                          2
newspaper_article                              2
search                                         2
converted                                      2
do_not_call                                    2
x_education_forums                             2
asymmetrique_activity_index                    3
asymmetrique_profile_index                     3
asymmetrique_profile_score                    10
asymmetrique_activity_score                   11
amount_missing                                14
Lead_Source                                   14
totalvisits                                   40
page_views_per_visit                         103
total_time_spent_on_website                 1635
dtype: int64
```

columns Related to Web Visits

In [53]:
```
fig, ax  = plt.subplots(3, figsize=(8,6))
sns.barplot(data=train_, x='totalvisits', y='converted',
            orient='h', palette='viridis',
            seed=2, ax=ax[0])
ax[0].set_title(f'Avg. Number of visits', loc='left', size=18)

sns.barplot(data=train_, x='total_time_spent_on_website', y='converted',
            orient='h', palette='viridis',
            seed=2, ax=ax[1])
ax[1].set_title(f'Avg. Time spent on website', loc='left', size=18)

sns.barplot(data=train_, x='page_views_per_visit', y='converted',
            orient='h', palette='viridis',
            seed=2, ax=ax[2])
ax[2].set_title(f'Avg. Page views per visit', loc='left', size=18)

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



In [54]:
```
fig, ax = plt.subplots(3,1, figsize=(8,6))
sns.boxplot(data=train_, x='totalvisits',
              ax=ax[0], palette='viridis')
ax[0].set_title('Total Visits', loc='left', size=16)

sns.boxplot(data=train_, x='total_time_spent_on_website',
              ax=ax[1], palette='viridis')
ax[1].set_title('Time spent on web', loc='left', size=16)

sns.boxplot(data=train_, x='page_views_per_visit',
              ax=ax[2], palette='viridis')
ax[2].set_title('Page views per visit', loc='left', size=16)

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



Insights:

There's a significant difference in conversion rate among both groups. Leads that convert more spent much more time on the website

5. Data Wrangling

Outlier treatments

Addressing outliers in TotalVisits and Page Views Per Visit is essential for model performance, particularly in Logistic Regression. Capping these variables at the 95th percentile is recommended for model stability and preventing inflated coefficients. It enhances model generalization in various classification models like Decision Trees, Random Forests, and Support Vector Machines.

Missing Values Statergies

- Numeric Columns (KNN Imputation): Utilizing KNNImputer for imputing missing values in Total Visits and Page Views Per Visit is a preferable choice over median, mean, or mode imputation. KNNImputer considers feature relationships, preserving data distribution, and handling multicollinearity effectively.

- Categorical Columns (Missing Category): Treating missing values as a separate category, rather than imputing with the mode, maintains data integrity, avoids biases, and improves model reliability and accuracy, especially considering the significant difference in conversion rate between leads with missing records and others.

5.1 Feature Engineer

apply all the insights discovered during EDA.

In [55]:
```
def eda_feature_engineering(df):
  # tags column
  df['tags'] = df['tags'].str.replace('|'.join(['invalid number','wrong number give
  df['tags'] = df['tags'].str.replace('|'.join(["In confusion whether part time or
  df['tags'] = df['tags'].str.replace("University not recognized","Not elegible")
  df['tags'] = df[df['tags'].notnull()].tags.apply(lambda x: 'Not elegible' if 'hol
  df['tags'] = df['tags'].str.replace('|'.join(["Interested in other courses", "Int
  df['tags'] = df['tags'].str.replace('|'.join(["Ringing","switched off"]),"Still n
  df['tags'] = df['tags'].str.replace('|'.join(["Want to take admission but has fin
  df['tags'] = df[df['tags'].notnull()].tags.apply(lambda x: 'Not elegible for the
  df['tags'] = df[df['tags'].notnull()].tags.apply(lambda x: 'Other' if x not in df

  # country and city
  indian_cities = ['Mumbai','Thane & Outskirts','Other Cities of Maharashtra','Tier
  df.loc[(df.country != 'India') & (df.city.isin(indian_cities)),'country'] = 'Indi
  df['country'] = df.loc[df['country'].notnull(),'country'].apply(lambda x: 'Other'

  # lead quality
  df['lead_quality'] = df['lead_quality'].fillna('Not Sure')

  # convert asymmetrique index columns in strings columns
  df[['asymmetrique_profile_index','asymmetrique_activity_index']] = df[['asymmetri

  # drop columns with unique values
  drop_cols = ['magazine','receive_more_updates_about_our_courses','update_me_on_su
               'get_updates_on_dm_content','i_agree_to_pay_the_amount_through_chequ
  df = df.drop(drop_cols, axis=1)

  #add amount_missing column
  df['amount_missing'] = df.isnull().sum(1)
  return df

eda_feature_engineering = FunctionTransformer(eda_feature_engineering)
```

5.2 Handling Outliers

In [56]:
```
def cap_outliers(df):
  """Replace outliers with the 95th percentile"""
  num_cols = ['totalvisits','page_views_per_visit','total_time_spent_on_website']
  df[num_cols[0]].apply(lambda x: df[num_cols[0]].quantile(.95) if x > df[num_cols[
  df[num_cols[1]].apply(lambda x: df[num_cols[1]].quantile(.95) if x > df[num_cols[
  df[num_cols[2]].apply(lambda x: df[num_cols[2]].quantile(.95) if x > df[num_cols[
  return df
```

```
cap_outliers = FunctionTransformer(cap_outliers);
```

5.3 Handling missing values and scaling columns for modeling

- 1. Apply OneHotEncoder to all the categorical columns.
- 2.

Apply StandardScaler to the numeric columns if there aren't binary.

In [57]:
```
cat_columns = ['lead_origin','lead_source','country','what_is_your_current_occupati
                'what_matters_most_to_you_in_choosing_a_course','tags','lead_qualit
                'city','last_notable_activity']

num_cols = ['totalvisits','page_views_per_visit','total_time_spent_on_website',
            'asymmetrique_activity_score','asymmetrique_profile_score','amount_miss

impute_knn = KNNImputer(n_neighbors=5)
impute_cons = SimpleImputer(strategy='constant', fill_value='Missing')
ohe = OneHotEncoder(handle_unknown='ignore')
sc = StandardScaler()

# Make pipelines for both type of columns treatments
pipe_cat = make_pipeline(impute_cons,ohe)
pipe_num = make_pipeline(sc,impute_knn)

impute_scale = make_column_transformer(
                                        (pipe_cat, cat_columns),
                                        (pipe_num,num_cols),
                                        remainder='drop'
                                            )
```

5.4 Saperate X and Y

In [58]:
```
X_train = train.drop('Converted',axis=1)
y_train = train.loc[:,'Converted']
```

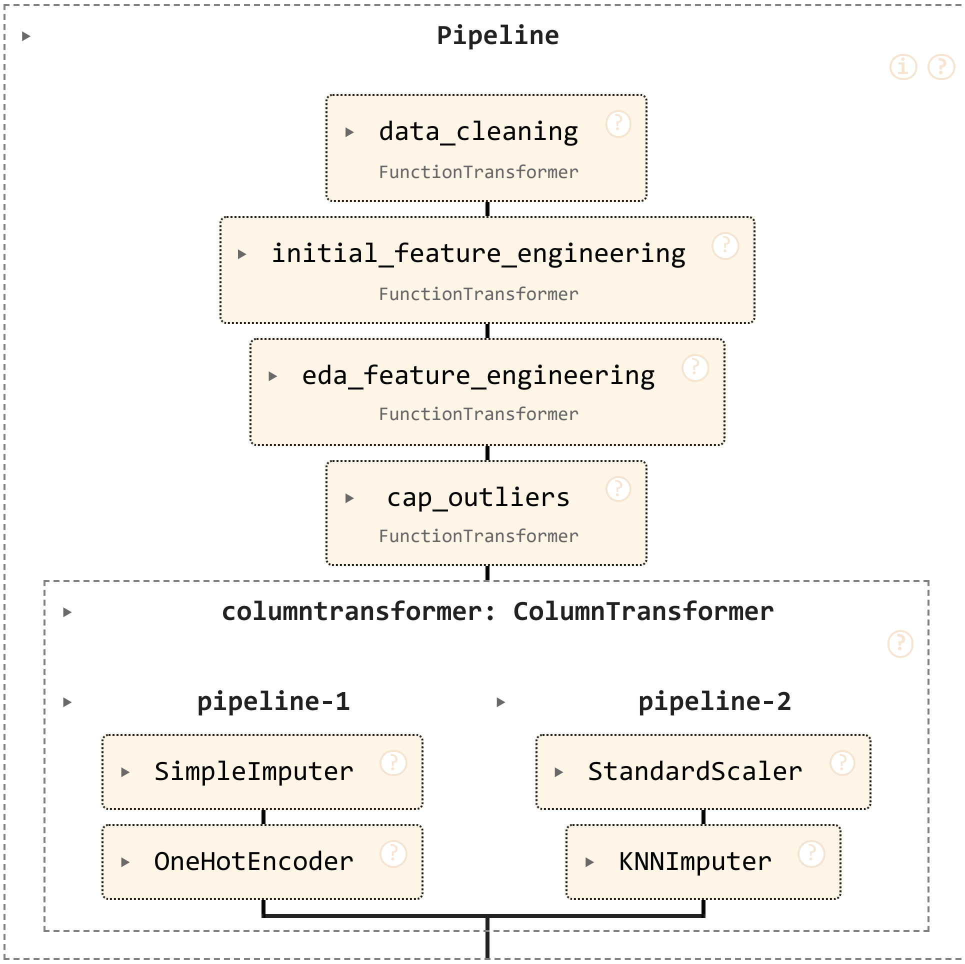
5.5 Create an entire pipeline for all preprocessing steps!¶

Creating a comprehensive preprocessing pipeline for ML is essential for consistency, efficiency, and reproducibility. It prevents data leakage, simplifies scaling, and integrates hyperparameter tuning seamlessly. Such a pipeline also aids in model deployment, enhancing performance, and maintaining a reliable ML workflow.

In [59]:
```
pipe = make_pipeline(
                    initial_clean,
                    initial_feature_engineering,
                    eda_feature_engineering,
                    cap_outliers,
                    impute_scale
              )
```

```python
# Let's see how it looks
pipe
```

Out[59]:



In [62]:
```python
X_train_pp = pipe.fit_transform(X_train)
```

6. Modeling

We'll start by exploring models for potential strong performance. First, we'll evaluate them using cross-validation with stratified folds to maintain class proportions.The goal is to identify promising models before fine-tuning hyperparameters.

Let's Remember Our Initial Target:

"The CEO, in particular, has given a ballpark of the target lead conversion rate to be around 80."

So, with this in mind, we can select our most important performance measure. In this case, we want to ensure that a high percentage of predicted leads convert to a customer, which means we're looking for a high precision score.

Does that mean we won't care about potential leads not detected (Low recall)?

Not at all. If we tune our models to optimize only for precision, we might be very accurate in their positive predictions but miss many actual positive cases. This translates into leaving money on the table—potential customers that won't convert.

Display function and SratifiedKFold

In [63]:
```
# Use stratified fold for ensure that we shuffle the dataset and conserve classes
skfold = StratifiedKFold(5, shuffle=True, random_state=12)

def display_scores(model,scores,pred):
  print(f'----------- {model} -----------')
  print('')
  print("------------------ Cross validation scores:")
  print("Scores:", scores)
  print("Mean:", scores.mean())
  print("Standard deviation:", scores.std())
  print('')
  print("--------------- Scores in the training set:")
  print("Precision:", precision_score(y_train,pred))
  print("Recall:", recall_score(y_train,pred))
  print("F1 score:", f1_score(y_train,pred))
  print("ROC - AUC score:", roc_auc_score(y_train,pred))
```

6.1 Logistic Regression

In [64]:
```
lr = LogisticRegression()
lr_scores = cross_val_score(lr, X_train_pp, y_train,
                            cv=skfold, scoring='f1')
lr.fit(X_train_pp,y_train)
lr_pred = lr.predict(X_train_pp)

# Precision and recall curve
lr_prec, lr_recall, lr_threshold = precision_recall_curve(y_train, lr_pred, pos_lab
lr_prdisplay = PrecisionRecallDisplay(precision=lr_prec, recall=lr_recall)

# Display Scores
display_scores('Logistic Regression',lr_scores,lr_pred)
```

```
----------- Logistic Regression -----------

------------------ Cross validation scores:
Scores: [0.91741472 0.91756272 0.91785714 0.91896705 0.93109541]
Mean: 0.9205794094986535
Standard deviation: 0.0052860620627966275

--------------- Scores in the training set:
Precision: 0.9382491827097712
Recall: 0.9066339066339066
F1 score: 0.9221706533380936
ROC - AUC score: 0.9346068498610871
```

6.2 Support Vector Machine

In [65]:
```
svc = SVC()
svc_scores = cross_val_score(svc, X_train_pp, y_train,
                             cv=skfold, scoring='f1')
svc.fit(X_train_pp, y_train)
svc_pred = svc.predict(X_train_pp)

# Precision and recall curve
svc_prec, svc_recall, svc_threshold = precision_recall_curve(y_train, svc_pred, pos
svc_prdisplay = PrecisionRecallDisplay(precision=svc_prec, recall=svc_recall)

# Display scores
display_scores('Support Vector Machine',svc_scores,svc_pred)
```

```
----------- Support Vector Machine -----------

------------------ Cross validation scores:
Scores: [0.91838565 0.92184725 0.9341637  0.91954023 0.93848858]
Mean: 0.9264850809036906
Standard deviation: 0.008226619600733422

--------------- Scores in the training set:
Precision: 0.9437074220150592
Recall: 0.9238329238329238
F1 score: 0.9336644200070947
ROC - AUC score: 0.9446371310778091
```

In [66]:
```
### 6.3 Decission Trees
```

In [67]:
```
tree = DecisionTreeClassifier(random_state = 7)
tree_scores = cross_val_score(tree, X_train_pp, y_train,
                              cv=skfold, scoring='f1')
tree.fit(X_train_pp, y_train)
tree_pred = tree.predict(X_train_pp)

# Precision and recall curve
tree_prec, tree_recall, tree_threshold = precision_recall_curve(y_train, tree_pred,
tree_prdisplay = PrecisionRecallDisplay(precision=tree_prec, recall=tree_recall)

# Display scores
display_scores('Decission Tree',tree_scores,tree_pred)
```

```
----------- Decission Tree -----------

------------------ Cross validation scores:
Scores: [0.89492119 0.89612676 0.89806678 0.89137931 0.89837746]
Mean: 0.8957743001598371
Standard deviation: 0.002537709857476365

--------------- Scores in the training set:
Precision: 0.9912434325744308
Recall: 0.9933309933309933
F1 score: 0.9922861150070126
ROC - AUC score: 0.9939140108631633
```

In [68]:
```
### 6.4 Random Forest
```

In [69]:
```
rf = RandomForestClassifier(random_state=10,
                            oob_score=True)
rf_scores = cross_val_score(rf, X_train_pp, y_train,
                            cv=skfold, scoring='f1')
rf.fit(X_train_pp, y_train)
rf_pred = rf.predict(X_train_pp)
rf_pred_proba = rf.predict_proba(X_train_pp)

# Precision and recall curve
rf_prec, rf_recall, rf_threshold = precision_recall_curve(y_train, rf_pred_proba[:,
rf_prdisplay = PrecisionRecallDisplay(precision=rf_prec, recall=rf_recall)

# Display scores
display_scores('Random Forest',rf_scores,rf_pred)
print('Oob score: ',rf.oob_score_)
```

```
----------- Random Forest -----------

------------------ Cross validation scores:
Scores: [0.91921005 0.91974752 0.93027361 0.9215859  0.93960924]
Mean: 0.9260852646703863
Standard deviation: 0.007850093308105268

--------------- Scores in the training set:
Precision: 0.9908995449772489
Recall: 0.9936819936819937
F1 score: 0.9922888187872415
ROC - AUC score: 0.9939794516065702
Oob score:  0.9422348484848485
```

6.5 Gradient Boosting

In [70]:
```
xg = GradientBoostingClassifier(random_state=11)
xg_scores = cross_val_score(xg, X_train_pp, y_train,
                            cv=skfold, scoring='f1')
xg.fit(X_train_pp, y_train)
xg_pred = xg.predict(X_train_pp)

# Precision and recall curve
xg_prec, xg_recall, xg_threshold = precision_recall_curve(y_train, xg_pred, pos_lab
xg_prdisplay = PrecisionRecallDisplay(precision=xg_prec, recall=xg_recall)

# Display scores
display_scores('Gradient Boosting',xg_scores,xg_pred)
```

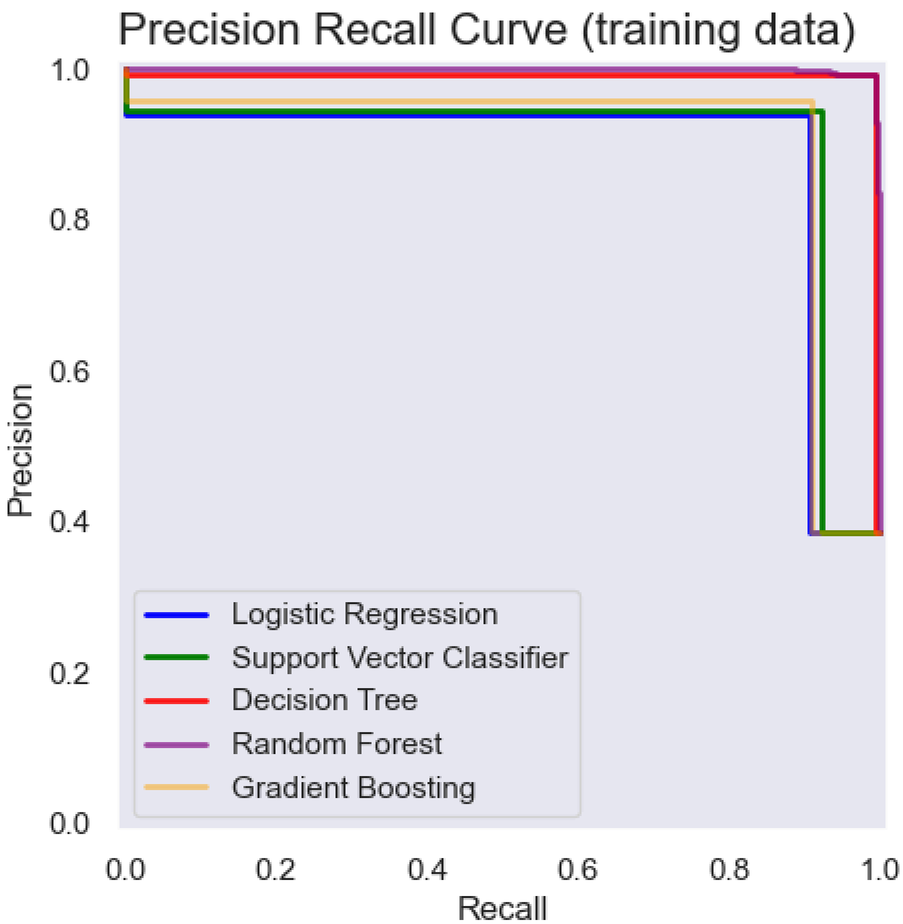
```
----------- Gradient Boosting -----------

------------------ Cross validation scores:
Scores: [0.91862568 0.92196007 0.9309417  0.92197309 0.94201606]
Mean: 0.927103321202558
Standard deviation: 0.008506063041899237

--------------- Scores in the training set:
Precision: 0.9571322985957132
Recall: 0.9090909090909091
F1 score: 0.9324932493249325
ROC - AUC score: 0.9417785604226282
```

7. Select the best model and tune them

7.1 Recall - Precision Curve for each model

In [71]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(8,5))
lr_prdisplay.plot(ax=ax, label='Logistic Regression', color='blue', linewidth=2)
svc_prdisplay.plot(ax=ax, label='Support Vector Classifier', color='green', linewid
tree_prdisplay.plot(ax=ax, label='Decision Tree', color='red', linewidth=2, alpha=.
rf_prdisplay.plot(ax=ax, label='Random Forest', color='purple', linewidth=2, alpha=
xg_prdisplay.plot(ax=ax, label='Gradient Boosting', color='orange', linewidth=2, al
plt.title('Precision Recall Curve (training data)', size=16, loc='left')
plt.show()
```



8.Make Our Prediction

At this point we are already:

1.

Completed the entire data preprocessing and exploration.#### 2.
We exclusively used the training dataset to eliminate any potential human bias#### 3. .
Additionally, we've incorporated all the preprocessing steps into a pipeline to prevent any data leak.

4. e.

Next, we selected the most promising models (without tuning) and applied cross-validation to assess their performan#### 5. ce. Following that, we fine-tuned those models using RandomizedSearchCV and identified the bee.st on

8.2 Apply al the preprocessing pipeline to the test dataset

In [72]:
```
X_test = test.drop(columns=['Converted'])  # More readable way to drop a column
y_test = test['Converted']  # Directly selecting the column
```

```
# Viewing the first row as a NumPy array
X_test.iloc[:1].to_numpy()
```

Out[72]:
```
array([['b4d86fa1-53d9-4a27-8d0c-f6603a562184', 634844, 'API', 'Google',
        'No', 'No', 2.0, 1551, 1.0, 'SMS Sent', 'India', nan, nan, nan,
        nan, 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', nan, nan,
        'No', 'No', nan, nan, '02.Medium', '02.Medium', 15.0, 15.0, 'No',
        'No', 'SMS Sent']], dtype=object)
```

In [73]:
```
# apply all the preprocessing steps to the test dataset
X_test_pp = pipe.transform(X_test)
X_test_pp.toarray()[:1]
```

Out[73]:
```
array([[ 1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  1.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  1.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  1.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  1.        ,  0.        ,  0.        ,
         0.        , -0.33742588, -0.62669473,  1.93925017,  0.50806717,
        -0.74681403,  1.33195148]])
```

10.2 Random Forest with hyperparameter tuned

In [74]:
```
print(X_train.dtypes)
from sklearn.preprocessing import OneHotEncoder
import pandas as pd

# Apply One-Hot Encoding
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)

# Ensure both datasets have the same columns
X_train, X_test = X_train.align(X_test, join="left", axis=1, fill_value=0)

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
for col in X_train.select_dtypes(include=['object']).columns:
    X_train[col] = encoder.fit_transform(X_train[col])
    X_test[col] = encoder.transform(X_test[col])

X_train = X_train.fillna(0)  # Replace NaN with 0
X_test = X_test.fillna(0)

rf_randomcv.fit(X_train, y_train)
```

```
Prospect ID                                       object
Lead Number                                        int64
Lead Origin                                       object
Lead Source                                       object
Do Not Email                                      object
Do Not Call                                       object
TotalVisits                                      float64
Total Time Spent on Website                        int64
Page Views Per Visit                             float64
Last Activity                                     object
Country                                           object
Specialization                                    object
How did you hear about X Education                object
What is your current occupation                   object
What matters most to you in choosing a course     object
Search                                            object
Magazine                                          object
Newspaper Article                                 object
X Education Forums                                object
Newspaper                                         object
Digital Advertisement                             object
Through Recommendations                           object
Receive More Updates About Our Courses            object
Tags                                              object
Lead Quality                                      object
Update me on Supply Chain Content                 object
Get updates on DM Content                         object
Lead Profile                                      object
City                                              object
Asymmetrique Activity Index                       object
Asymmetrique Profile Index                        object
Asymmetrique Activity Score                      float64
Asymmetrique Profile Score                       float64
I agree to pay the amount through cheque          object
A free copy of Mastering The Interview            object
Last Notable Activity                             object
dtype: object
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[74], line 22
     19 X_train = X_train.fillna(0)  # Replace NaN with 0
     20 X_test = X_test.fillna(0)
---> 22 rf_randomcv.fit(X_train, y_train)

NameError: name 'rf_randomcv' is not defined
```

In [ ]:

```
rf_rcv_pred = rf_randomcv.predict(X_test_pp)
print("Precision:", precision_score(y_test,rf_rcv_pred))
print("Recall:", recall_score(y_test,rf_rcv_pred))
print("F1 score:", f1_score(y_test,rf_rcv_pred))
print("ROC - AUC score:", roc_auc_score(y_test,rf_rcv_pred))
```

10.3 Random Forest without hyperparameter tuned

In [ ]:
```
rf_pred_test = rf.predict(X_test_pp)
print("Precision:", precision_score(y_test,rf_pred_test))
print("Recall:", recall_score(y_test,rf_pred_test))
print("F1 score:", f1_score(y_test,rf_pred_test))
print("ROC - AUC score:", roc_auc_score(y_test,rf_pred_test))
```

10.4 Let's plot the confussion matrix for both models!

In [ ]:
```
fig, ax = plt.subplots(1, 2, figsize=(10, 4))

# Random Forest tunned
cm1 = confusion_matrix(y_test, rf_rcv_pred)
sns.heatmap(cm1, annot=True, fmt = 'd', cmap='Greens', ax = ax[0], cbar=False)
ax[0].xaxis.set_ticklabels(['Not converted', 'Converted'])
ax[0].yaxis.set_ticklabels(['Not converted', 'Converted'])
ax[0].set_title('RF with hyperparameters tuning', loc='left')
ax[0].set_xlabel('Predicted')
ax[0].set_ylabel('True')

# Random Forest without tuning
cm2 = confusion_matrix(y_test, rf_pred_test)
sns.heatmap(cm2, annot=True, fmt='d', cmap='Blues', ax=ax[1], cbar=False)
ax[1].xaxis.set_ticklabels(['Not converted', 'Converted'])
ax[1].yaxis.set_ticklabels(['Not converted', 'Converted'])
ax[1].set_title('RF without hyperparameters tuning', loc='left')
ax[1].set_xlabel('Predicted')
ax[1].set_ylabel('True')

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

In [ ]:
```
import matplotlib.pyplot as plt

# Plot a histogram of lead scores
plt.hist(lead_scoring, bins=20, edgecolor='black')
plt.xlabel("Lead Score")
plt.ylabel("Frequency")
plt.title("Distribution of Lead Scores")
plt.show()
```

In [ ]:
```
plt.scatter(lead_prediction, lead_scoring, alpha=0.5)
plt.xlabel("Lead Prediction (0 or 1)")
plt.ylabel("Lead Score")
plt.title("Lead Prediction vs. Lead Score")
plt.show()
```

Submission

- Class predictions in the left, and probabilities to convert into a customer on the right.

In [ ]:
```
import numpy as np
```

```
# Assuming rf_randomcv is a trained RandomForest model and X_test_pp is preprocesse
lead_scoring = rf_randomcv.predict_proba(X_test_pp)[:, 1]  # Getting probabilities
lead_prediction = rf_randomcv.predict(X_test_pp)  # Getting predictions

# Combining predictions and probabilities
results = np.round(np.c_[lead_prediction, lead_scoring], 2)

# Display the first 10 rows
print(results[:10])
```

Conclusion

In summary, our data science project focused on fine-tuning lead scoring for X Education. We aimed to exceed an 80% precision goal, which we not only met but exceeded. Throughout our journey, we identified key factors like phone interactions, referrals, and online engagement that strongly correlated with lead conversion, leading to actionable strategies.

One notable achievement was the development of an automated lead scoring algorithm that not only improved lead assessment precision but also streamlined operational efficiency. By targeting promising leads, X Education could reduce sales team costs significantly.

Our journey involved thorough data exploration, preprocessing, and model development, ensuring consistency and mitigating bias. We systematically evaluated models, with the tuned Random Forest model achieving an impressive F1 score of 0.9287 and a precision score of 0.9527 on the test dataset.

This data-driven journey provides X Education with actionable insights to enhance efficiency and revenue growth, positioning the company for a transformative phase.

If you've read until here, thank you. I hope you found this information helpful and interesting in some way. Your feedback is greatly appreciated. Best regards.