Lead Scoring Case Study

Problem Statement

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. The typical lead conversion rate at X education is around 30%.

Now, although X Education gets a lot of leads, its lead conversion rate is very poor. For example, if, say, they acquire 100 leads in a day, only about 30 of them are converted. To make this process more efficient, the company wishes to identify the most potential leads, also known as 'Hot Leads'. If they successfully identify this set of leads, the lead conversion rate should go up as the sales team will now be focusing more on communicating with the potential leads rather than making calls to everyone.

There are a lot of leads generated in the initial stage (top) but only a few of them come out as paying customers from the bottom. In the middle stage, we need to nurture the potential leads well (i.e. educating the leads about the product, constantly communicating etc.) in order to get a higher lead conversion.

X Education has appointed me to help them select the most promising leads, i.e. the leads that are most likely to convert into paying customers. The company requires me to build a model wherein I'll 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%.

Data

You have been provided with a leads dataset from the past with around 9000 data points. This dataset consists of various attributes such as Lead Source, Total Time Spent on Website, Total Visits, Last Activity, etc. which may or may not be useful in ultimately deciding whether a lead will be converted or not. The target variable, in this case, is the column 'Converted' which tells whether a past lead was converted or not wherein 1 means it was converted and 0 means it wasn't converted. You can learn more about the dataset from the data dictionary provided in the zip folder at the end of the page. Another thing that you also need to check out for are the levels present in the categorical variables. Many of the categorical variables have a level called 'Select' which needs to be handled because it is as good as a null value (think why?).

Table of Contents

```
# basic libraries to work on the dataframe
import pandas as pd
import numpy as np
# data Visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns

# libraries
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')

#Increasing the columns views limit
pd.options.display.max_columns = None
pd.options.display.max_rows = 150
pd.options.display.float_format = '{:.2f}'.format
```

Reading and Understanding Data

```
#Reading the data file using pandas
df = pd.read_csv('Leads.csv')

df.head()
```

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.00	0	0.00	Page Visited on Website
1	2a272436-5132-4136-86fa-dcc88c88f482	660728	API	Organic Search	No	No	0	5.00	674	2.50	Email Opened
2	8cc8c611-a219-4f35-ad23-fdfd2656bd8a	660727	Landing Page Submission	Direct Traffic	No	No	1	2.00	1532	2.00	Email Opened
3	0cc2df48-7cf4-4e39-9de9-19797f9b38cc	660719	Landing Page Submission	Direct Traffic	No	No	0	1.00	305	1.00	Unreachable

	Prospect ID	Lead Number	Lead Origin	Lead Source	Do Not Email	Do Not Call	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Last Activity
4	3256f628-e534-4826-9d63-4a8b88782852	660681	Landing Page Submission	Google	No	No	1	2.00	1428	1.00	Converted to Lead

```
# check the shape of the dataset
df.shape
```

```
(9240, 37)
```

```
# check statistics for numerical columns
df.describe()
```

	Lead Number	Converted	TotalVisits	Total Time Spent on Website	Page Views Per Visit	Asymmetrique Activity Score	Asymmetrique Profile Score
count	9240.00	9240.00	9103.00	9240.00	9103.00	5022.00	5022.00
mean	617188.44	0.39	3.45	487.70	2.36	14.31	16.34
std	23406.00	0.49	4.85	548.02	2.16	1.39	1.81
min	579533.00	0.00	0.00	0.00	0.00	7.00	11.00
25%	596484.50	0.00	1.00	12.00	1.00	14.00	15.00
50%	615479.00	0.00	3.00	248.00	2.00	14.00	16.00
75%	637387.25	1.00	5.00	936.00	3.00	15.00	18.00
max	660737.00	1.00	251.00	2272.00	55.00	18.00	20.00

```
# check whether there are any duplicates
df.duplicated().sum()
```

```
0
```

```
#Lets have a look at all the columns, their datatypes and also get an idea of null valu
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
```

Observations

- A large number of columns have null values. Those columns should ideally be dropped
- `Prospect ID` and `Lead Number` both serve the same purpose. They are both unique identifiers. We will drop `Prospect ID`
- Column names are just too long. We will modify the column names
- Few categorical columns have "Select" in their entries. Those select are essentially null values because Select appears when someone does not select anything from the dropdown

Data Cleaning

Rename column names

- Long column names make analysis tiring as one has to always refer to column names. Also has impact on charts created later on
- Ideally, we should follow python's preferred Snakecase nomenclature

```
# change nomenclature to snakecase
df.columns = df.columns.str.replace(' ', '_').str.lower()

# test
df.columns
```

```
Index(['prospect_id', 'lead_number', 'lead_origin', 'lead_source',
       'do_not_email', 'do_not_call', 'converted', 'totalvisits',
       'total_time_spent_on_website', 'page_views_per_visit', 'last_activity',
       'country', 'specialization', 'how_did_you_hear_about_x_education',
       'what_is_your_current_occupation',
       'what_matters_most_to_you_in_choosing_a_course', 'search', 'magazine',
       'newspaper_article', 'x_education_forums', 'newspaper',
       'digital_advertisement', 'through_recommendations',
       'receive_more_updates_about_our_courses', 'tags', 'lead_quality',
       'update_me_on_supply_chain_content', 'get_updates_on_dm_content',
       'lead_profile', 'city', 'asymmetrique_activity_index',
       'asymmetrique_profile_index', 'asymmetrique_activity_score',
       'asymmetrique_profile_score',
       'i_agree_to_pay_the_amount_through_cheque',
       'a_free_copy_of_mastering_the_interview', 'last_notable_activity'],
      dtype='object')
```

```
# shorten column names
df.rename(columns = {'totalvisits': 'total_visits', 'total_time_spent_on_website': 'tim
                     'how_did_you_hear_about_x_education': 'source', 'what_is_your_curre
                     'what_matters_most_to_you_in_choosing_a_course' : 'course_selection
                     'receive_more_updates_about_our_courses': 'courses_updates',
                      'update_me_on_supply_chain_content': 'supply_chain_content_updates
                     'get_updates_on_dm_content': 'dm_content_updates',
                     'i_agree_to_pay_the_amount_through_cheque': 'cheque_payment',
                     'a_free_copy_of_mastering_the_interview': 'mastering_interview'}, i

df.head(1)
```

	prospect_id	lead_number	lead_origin	lead_source	do_not_email	do_not_call	converted	total_visits	time_on_we
0	7927b2df-8bba-4d29-b9a2-b6e0beafe620	660737	API	Olark Chat	No	No	0	0.00	

Drop `prospect_id` column

```
df.drop('prospect_id', axis = 1, inplace = True)
```

Replace "Select" category with null values

```
# Select all non-numeric columns
df_obj = df.select_dtypes(include='object')

# Find out columns that have "Select"
s = lambda x: x.str.contains('Select', na=False)
l = df_obj.columns[df_obj.apply(s).any()].tolist()
print (l)
```

```
['specialization', 'source', 'lead_profile', 'city']
```

There are 4 columns that contains `Select`, which are effectively null values. We are going to make that change

```
# select all the columns that have a "Select" entry
sel_cols = ['specialization', 'source', 'lead_profile', 'city']

# replace values
df[sel_cols] = df[sel_cols].replace('Select', np.NaN)
```

Handle null values and sales generated columns

- Given there are a number of columns with very high number of null entries, let's calculate the percentage of null values in each column, and take a decision from there.
- Furthermore, we can also drop Sales generated columns because those are the data entries that are made after the sales team has connected with the student. Those data have no bearing to the purpose of our model ie. providing lead score. The columns are
    - `tags`
    - `lead_quality`
    - all `asymmetrique` columns
    - `last_activity`
    - `last_notable_activity`

```
# Calculate percentage of null values for each column
(df.isnull().sum() / df.shape[0]) * 100
```

```
lead_number                       0.00
lead_origin                       0.00
lead_source                       0.39
do_not_email                      0.00
do_not_call                       0.00
converted                         0.00
```

```
total_visits                      1.48
time_on_website                   0.00
page_views_per_visit              1.48
last_activity                     1.11
country                          26.63
specialization                   36.58
source                           78.46
occupation                       29.11
course_selection_reason          29.32
search                            0.00
magazine                          0.00
newspaper_article                 0.00
x_education_forums                0.00
newspaper                         0.00
digital_advertisement             0.00
through_recommendations           0.00
courses_updates                   0.00
tags                             36.29
lead_quality                     51.59
supply_chain_content_updates      0.00
dm_content_updates                0.00
lead_profile                     74.19
city                             39.71
asymmetrique_activity_index      45.65
asymmetrique_profile_index       45.65
asymmetrique_activity_score      45.65
asymmetrique_profile_score       45.65
cheque_payment                    0.00
mastering_interview               0.00
last_notable_activity             0.00
dtype: float64
```

Observation: As can be seen, there are quite a few columns with high number of missing data. Since there are no ways to get data back from reliable sources, we can drop all those columns that have missing values > 40%

Drop columns that have null values > 40% or Sales generated columns

```
df.drop(['source', 'lead_quality', 'lead_profile', 'asymmetrique_activity_index',
                      'asymmetrique_profile_index', 'asymmetrique_activity_score', 'asy
        'tags', 'last_activity', 'last_notable_activity'],
        axis = 1, inplace = True)

df.head(1)
```

	lead_number	lead_origin	lead_source	do_not_email	do_not_call	converted	total_visits	time_on_website	page_view
0	660737	API	Olark Chat	No	No	0	0.00	0	

```
# Lets look at what are we left with
# Calculate percentage of null values for each column
```

```
(df.isnull().sum() / df.shape[0]) * 100
```

```
lead_number                      0.00
lead_origin                      0.00
lead_source                      0.39
do_not_email                     0.00
do_not_call                      0.00
converted                        0.00
total_visits                     1.48
time_on_website                  0.00
page_views_per_visit             1.48
country                         26.63
specialization                  36.58
occupation                      29.11
course_selection_reason         29.32
search                           0.00
magazine                         0.00
newspaper_article                0.00
x_education_forums               0.00
newspaper                        0.00
digital_advertisement            0.00
through_recommendations          0.00
courses_updates                  0.00
supply_chain_content_updates     0.00
dm_content_updates               0.00
city                            39.71
cheque_payment                   0.00
mastering_interview              0.00
dtype: float64
```

Observations

There are five columns that still have high null values: `country`, `specialization`, `occupation`, `course_selection_reason`, and `city`. We will look at them individually to see what can be done

`country` column

```
df.country.value_counts(normalize = True, dropna = False) * 100
```

```
India                   70.26
NaN                     26.63
United States            0.75
United Arab Emirates     0.57
Singapore                0.26
Saudi Arabia             0.23
United Kingdom           0.16
Australia                0.14
Qatar                    0.11
Bahrain                  0.08
Hong Kong                0.08
France                   0.06
```

```
Oman                     0.06
unknown                  0.05
South Africa             0.04
Nigeria                  0.04
Germany                  0.04
Kuwait                   0.04
Canada                   0.04
Sweden                   0.03
Bangladesh               0.02
Italy                    0.02
Belgium                  0.02
Ghana                    0.02
Uganda                   0.02
China                    0.02
Asia/Pacific Region      0.02
Philippines              0.02
Netherlands              0.02
Switzerland              0.01
Malaysia                 0.01
Kenya                    0.01
Liberia                  0.01
Sri Lanka                0.01
Vietnam                  0.01
Tanzania                 0.01
Denmark                  0.01
Indonesia                0.01
Russia                   0.01
Name: country, dtype: float64
```

Observation
The distribution of the data is very heavily skewed, with India + null values = 97% of the total. It is safe to drop this column.

```
df.drop('country', axis = 1, inplace = True)
```

`course_selection_reason` column

```
df.course_selection_reason.value_counts(normalize = True, dropna = False) * 100
```

```
Better Career Prospects      70.65
NaN                          29.32
Flexibility & Convenience     0.02
Other                         0.01
Name: course_selection_reason, dtype: float64
```

Observation
The distribution of the data is very heavily skewed, with Better career prospects + null values = approx 100% of the total. It is safe to drop this column.

```
df.drop('course_selection_reason', axis = 1, inplace = True)
```

occupation column

```
df.occupation.value_counts(normalize = True, dropna = False) * 100
```

```
Unemployed              60.61
NaN                     29.11
Working Professional     7.64
Student                  2.27
Other                    0.17
Housewife                0.11
Businessman              0.09
Name: occupation, dtype: float64
```

Observation
For occupation, we can first combine categories, and then impute proportionally to maintain the distribution and not introduce bias

```
# combine low representing categories
df.loc[(df.occupation == 'Student') | (df.occupation == 'Other') | (df.occupation == 'H
       (df.occupation == 'Businessman') , 'occupation'] = 'Student and Others'
```

```
df.occupation.value_counts(normalize = True) * 100
```

```
Unemployed              85.50
Working Professional    10.78
Student and Others       3.73
Name: occupation, dtype: float64
```

```
# impute proportionately
df['occupation'] = df.occupation.fillna(pd.Series(np.random.choice(['Unemployed', 'Work
                                                                    'Student and Others
                                                                   p = [0.8550, 0.1078,
```

specialization column

```
df.specialization.value_counts(normalize = True, dropna = False) * 100
```

```
NaN                                  36.58
Finance Management                   10.56
Human Resource Management             9.18
Marketing Management                  9.07
Operations Management                 5.44
Business Administration               4.36
IT Projects Management                3.96
Supply Chain Management               3.78
Banking, Investment And Insurance     3.66
```

```
Media and Advertising              2.20
Travel and Tourism                 2.20
International Business             1.93
Healthcare Management              1.72
Hospitality Management             1.23
E-COMMERCE                         1.21
Retail Management                  1.08
Rural and Agribusiness             0.79
E-Business                         0.62
Services Excellence                0.43
Name: specialization, dtype: float64
```

Observation

For specialization, we can first combine categories based on the course type, and then impute proportionally to maintain the distribution and not introduce bias

```
# categorize all management courses
df.loc[(df.specialization == 'Finance Management') | (df.specialization == 'Human Resou
       (df.specialization == 'Marketing Management') |  (df.specialization == 'Operatic
       (df.specialization == 'IT Projects Management') | (df.specialization == 'Supply
       (df.specialization == 'Healthcare Management') | (df.specialization == 'Hospital
       (df.specialization == 'Retail Management') , 'specialization'] = 'Management Spe

# categorize all busines courses
df.loc[(df.specialization == 'Business Administration') | (df.specialization == 'Interr
       (df.specialization == 'Rural and Agribusiness') | (df.specialization == 'E-Busir
        , 'specialization'] = 'Business Specializations'

# categorize all industry courses
df.loc[(df.specialization == 'Banking, Investment And Insurance') | (df.specialization
       (df.specialization == 'Travel and Tourism') | (df.specialization == 'Services Ex
       (df.specialization == 'E-COMMERCE'), 'specialization'] = 'Industry Specializatic
```

```
df.specialization.value_counts(normalize = True) * 100
```

```
Management Specializations    72.58
Industry Specializations      15.29
Business Specializations      12.13
Name: specialization, dtype: float64
```

```
# impute proportionately
df['specialization'] = df.specialization.fillna(pd.Series(np.random.choice(['Management
                                                      'Business Specializations', 'Indust
                                                                     p = [0.7258, 0.1213,
```

city column

```
df.city.value_counts(normalize = True, dropna = False) * 100
```

```
NaN                           39.71
Mumbai                        34.87
Thane & Outskirts              8.14
Other Cities                   7.42
Other Cities of Maharashtra    4.95
Other Metro Cities             4.11
Tier II Cities                 0.80
Name: city, dtype: float64
```

Observations We will categorize cities based on logical decisions and impute proportionately

```
# categorize all non-mumbai, but Maharashtra cities
df.loc[(df.city == 'Thane & Outskirts') | (df.city == 'Other Cities of Maharashtra'),
       'city'] = 'Non-Mumbai Maharashtra Cities'

# categorize all other cities
df.loc[(df.city == 'Other Cities') | (df.city == 'Other Metro Cities') | (df.city == 'T
       'city'] = 'Non-Maharashtra Cities'
```

```
df.city.value_counts(normalize = True) * 100
```

```
Mumbai                           57.84
Non-Mumbai Maharashtra Cities    21.70
Non-Maharashtra Cities           20.46
Name: city, dtype: float64
```

```
# impute proportionately
df['city'] = df.city.fillna(pd.Series(np.random.choice(['Mumbai', 'Non-Mumbai Maharasht
                                                         'Non-Maharashtra Ci
                                                        p = [0.5784, 0.2170,
```

Handle categorical columns with low number of missing values and low representation of categories

In this step, we will go through the rest of the categorical columns one by one and

- Merge categories that have low representation
- Impute the missing values

```
(df.isnull().sum() / df.shape[0]) * 100
```

```
lead_number                    0.00
lead_origin                    0.00
lead_source                    0.39
do_not_email                   0.00
do_not_call                    0.00
converted                      0.00
total_visits                   1.48
```

```
time_on_website                 0.00
page_views_per_visit            1.48
specialization                  0.00
occupation                      0.00
search                          0.00
magazine                        0.00
newspaper_article               0.00
x_education_forums              0.00
newspaper                       0.00
digital_advertisement           0.00
through_recommendations         0.00
courses_updates                 0.00
supply_chain_content_updates    0.00
dm_content_updates              0.00
city                            0.00
cheque_payment                  0.00
mastering_interview             0.00
dtype: float64
```

```
# determine unique values for all object datatype columns
for k, v in df.select_dtypes(include='object').nunique().to_dict().items():
    print('{} = {}'.format(k,v))
```

```
lead_origin = 5
lead_source = 21
do_not_email = 2
do_not_call = 2
specialization = 3
occupation = 3
search = 2
magazine = 1
newspaper_article = 2
x_education_forums = 2
newspaper = 2
digital_advertisement = 2
through_recommendations = 2
courses_updates = 1
supply_chain_content_updates = 1
dm_content_updates = 1
city = 3
cheque_payment = 1
mastering_interview = 2
```

Observation

As can be seen from the above output, the categorical columns (i.e. number of unique values > 2) are:

- `lead_origin`

- lead_source

lead_origin column

```
df.lead_origin.value_counts(normalize = True, dropna = False) * 100
```

```
Landing Page Submission    52.88
API                        38.74
Lead Add Form               7.77
Lead Import                 0.60
Quick Add Form              0.01
Name: lead_origin, dtype: float64
```

```
#There are a lot of smaller values which will not be used as definitive factors, lets g
df.loc[(df.lead_origin == 'Lead Import') | (df.lead_origin == 'Quick Add Form') | (df.l
       , 'lead_origin'] = 'Lead Add Form and Others'
```

lead_source column

```
df.lead_source.value_counts(normalize = True, dropna = False) * 100
```

```
Google              31.04
Direct Traffic      27.52
Olark Chat          18.99
Organic Search      12.49
Reference            5.78
Welingak Website     1.54
Referral Sites       1.35
Facebook             0.60
NaN                  0.39
bing                 0.06
google               0.05
Click2call           0.04
Press_Release        0.02
Live Chat            0.02
Social Media         0.02
youtubechannel       0.01
Pay per Click Ads    0.01
testone              0.01
NC_EDM               0.01
welearnblog_Home     0.01
blog                 0.01
WeLearn              0.01
Name: lead_source, dtype: float64
```

```
# Lets impute the missing values with the mode of data i.e. clearly 'Google'
df.lead_source.fillna(df.lead_source.mode()[0], inplace=True)
```

```
#There are a lot of smaller values which will not be used as definitive factors, lets g
df['lead_source'] = df['lead_source'].apply(lambda x: x if
                                            ((x== 'Google') | (x=='Direct Traffic') | (
                                             (x=='Organic Search') | (x=='Reference'))
                                            else 'Other Social Sites')
```

Handle Binary columns

- Drop those columns that have significant data imbalance
- Drop all those columns that have only 1 unique entry

```
# determine unique values
for k, v in df.select_dtypes(include='object').nunique().to_dict().items():
    print('{} = {}'.format(k,v))
```

```
lead_origin = 3
lead_source = 6
do_not_email = 2
do_not_call = 2
specialization = 3
occupation = 3
search = 2
magazine = 1
newspaper_article = 2
x_education_forums = 2
newspaper = 2
digital_advertisement = 2
through_recommendations = 2
courses_updates = 1
supply_chain_content_updates = 1
dm_content_updates = 1
city = 3
cheque_payment = 1
mastering_interview = 2
```

Observation

- The following columns can be dropped as they have just 1 unique values
  - `magazine`
  - `course_updates`
  - `supply_chain_content_updates`
  - `dm_content_updates`
  - `cheque_payment`

Let's now check the data imbalance for the rest of the columns

```
# select rest of the binary columns in a new dataframe
df_bin = df[['do_not_email', 'do_not_call', 'search', 'newspaper_article', 'x_education
           'newspaper', 'digital_advertisement', 'through_recommendations', 'mastering_

# see value counts for each of the columns
for i in df_bin.columns:
    x = (df_bin[i].value_counts(normalize = True)) * 100
    print(x)
    print()
```

```
No     92.06
Yes     7.94
Name: do_not_email, dtype: float64

No     99.98
Yes     0.02
Name: do_not_call, dtype: float64

No     99.85
Yes     0.15
Name: search, dtype: float64

No     99.98
Yes     0.02
Name: newspaper_article, dtype: float64

No     99.99
Yes     0.01
Name: x_education_forums, dtype: float64

No     99.99
Yes     0.01
Name: newspaper, dtype: float64

No     99.96
Yes     0.04
Name: digital_advertisement, dtype: float64

No     99.92
Yes     0.08
Name: through_recommendations, dtype: float64

No     68.74
```

```
Yes    31.26
Name: mastering_interview, dtype: float64
```

Observations
Because of heavy data imbalance, we can drop the following columns as well

- `do_not_call`
- `search`
- `newspaper_article`
- `x_education_forums`
- `newspaper`
- `digital_advertisement`
- `through_recommendations`

```
drop_bin = ['do_not_call', 'search', 'newspaper_article', 'x_education_forums',
            'newspaper', 'digital_advertisement', 'through_recommendations', 'magazine',
            'supply_chain_content_updates', 'dm_content_updates', 'cheque_payment']

df.drop(drop_bin, axis = 1, inplace = True)
```

Handle Numerical columns

`lead_number` column: change datatype

`lead_number` column is a unique identifier for each leads. Therefore, aggregations won't be of any relevance. We should change it to object

```
df.lead_number = df.lead_number.astype('object')
```

`total_visits` column

For this column, we need to handle the missing values, and can convert the datatype to integer since visits can't be decimal

```
df.total_visits.fillna(df.total_visits.median(), inplace=True)
df.total_visits = df.total_visits.astype('int')
```

`page_views_per_visit` column

Handle missing values

```
df.page_views_per_visit.fillna(df.page_views_per_visit.median(), inplace=True)
```

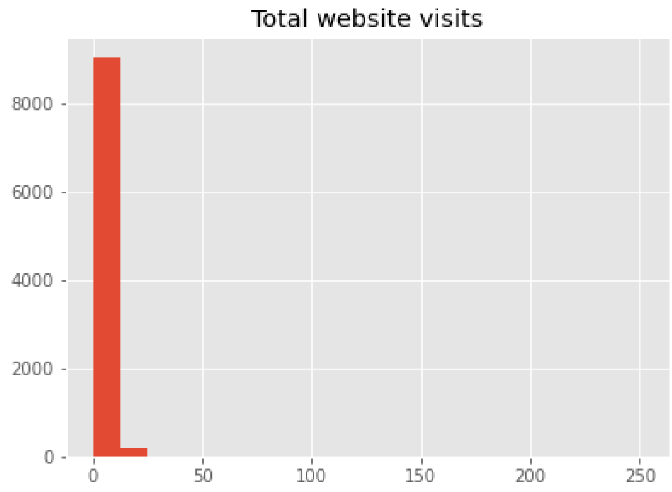
```
df.info()
```

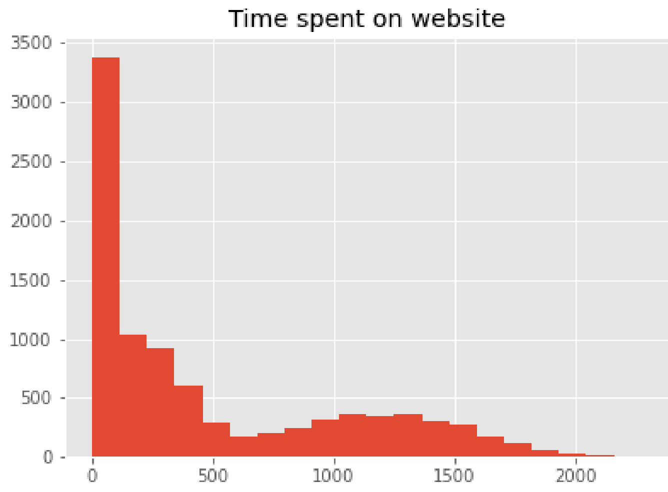
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9240 entries, 0 to 9239
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   lead_number           9240 non-null   object 
 1   lead_origin           9240 non-null   object 
 2   lead_source           9240 non-null   object 
 3   do_not_email          9240 non-null   object 
 4   converted             9240 non-null   int64  
 5   total_visits          9240 non-null   int32  
 6   time_on_website       9240 non-null   int64  
 7   page_views_per_visit  9240 non-null   float64
 8   specialization        9240 non-null   object 
 9   occupation            9240 non-null   object 
 10  city                  9240 non-null   object 
 11  mastering_interview   9240 non-null   object 
dtypes: float64(1), int32(1), int64(2), object(8)
memory usage: 830.3+ KB
```

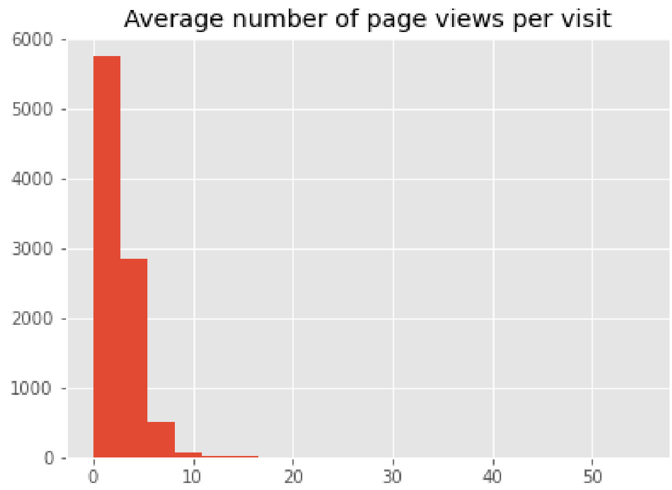
Exploratory Data Analysis

Numerical columns

```
# Set style
plt.style.use('ggplot')

# See distribution of each of these columns
fig = plt.figure(figsize = (14, 10))
plt.subplot(2, 2, 1)
plt.hist(df.total_visits, bins = 20)
plt.title('Total website visits')

plt.subplot(2, 2, 2)
plt.hist(df.time_on_website, bins = 20)
plt.title('Time spent on website')

plt.subplot(2, 2, 3)
plt.hist(df.page_views_per_visit, bins = 20)
plt.title('Average number of page views per visit')

plt.show()
```





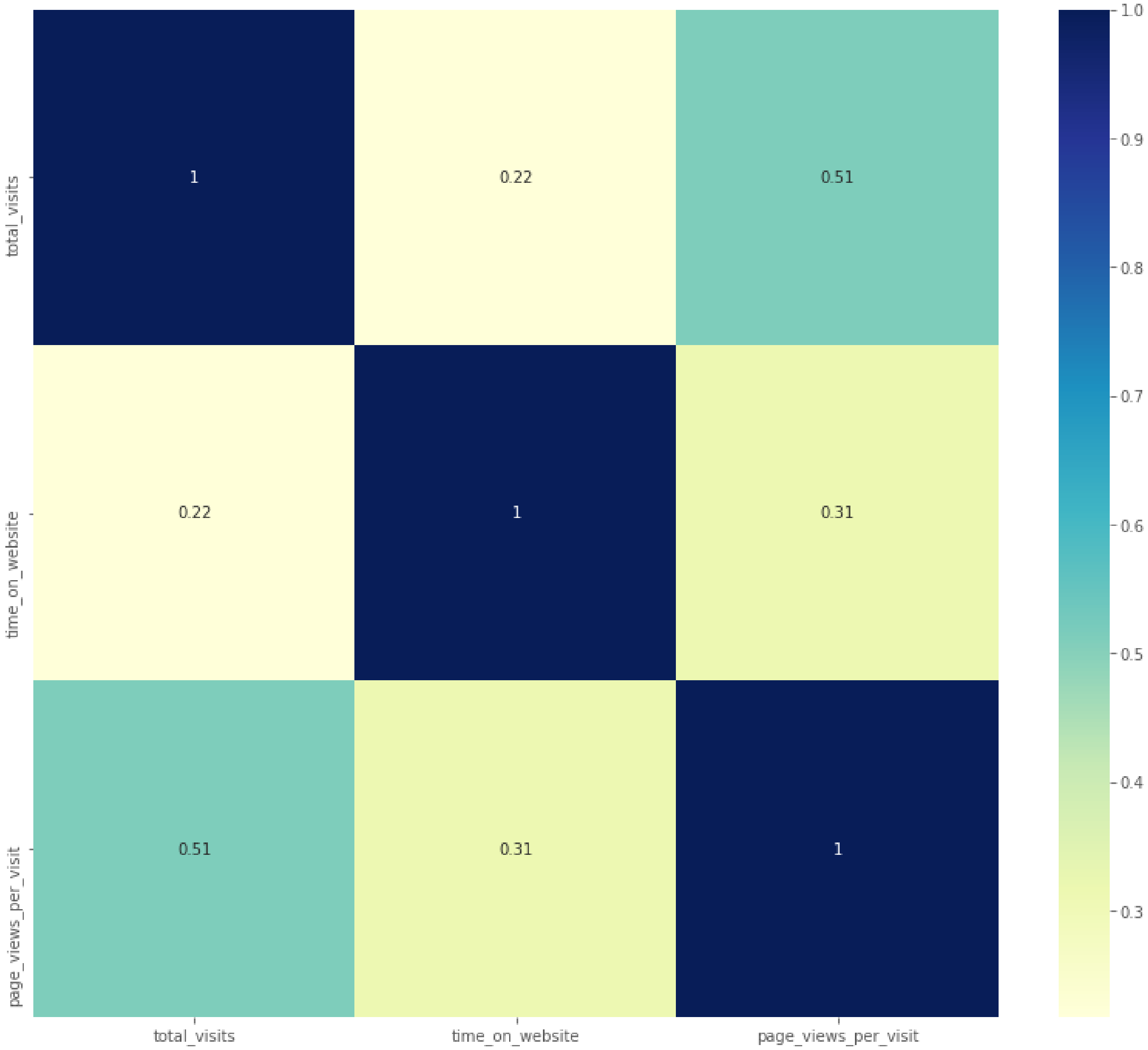


Observations

- High peaks and skewed data. There might be a possibility of outliers. We will check them next

Heatmap

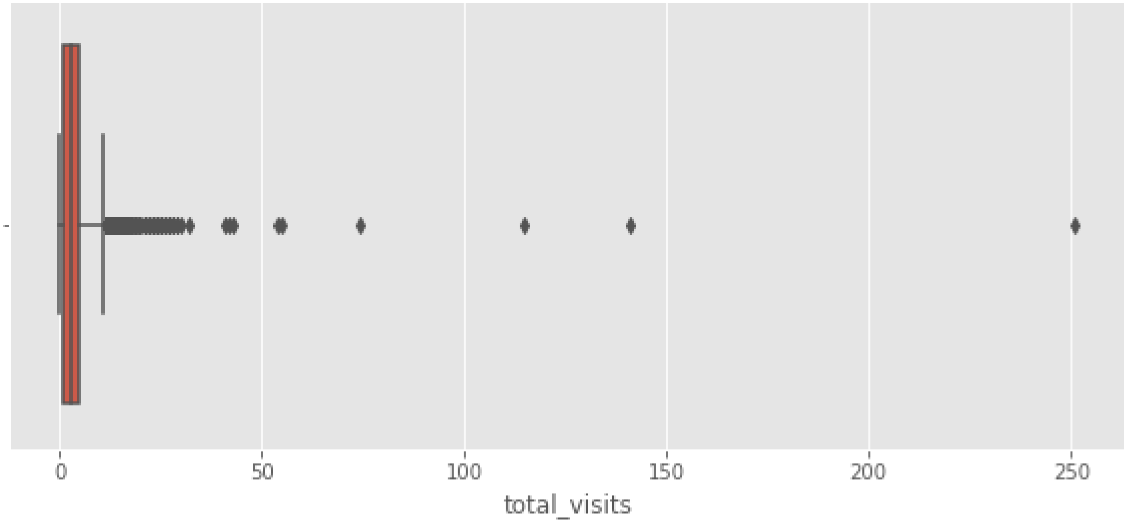
```
plt.figure(figsize = (14,12))
sns.heatmap(df[['total_visits', 'time_on_website', 'page_views_per_visit']].corr(), cma
plt.show()
```

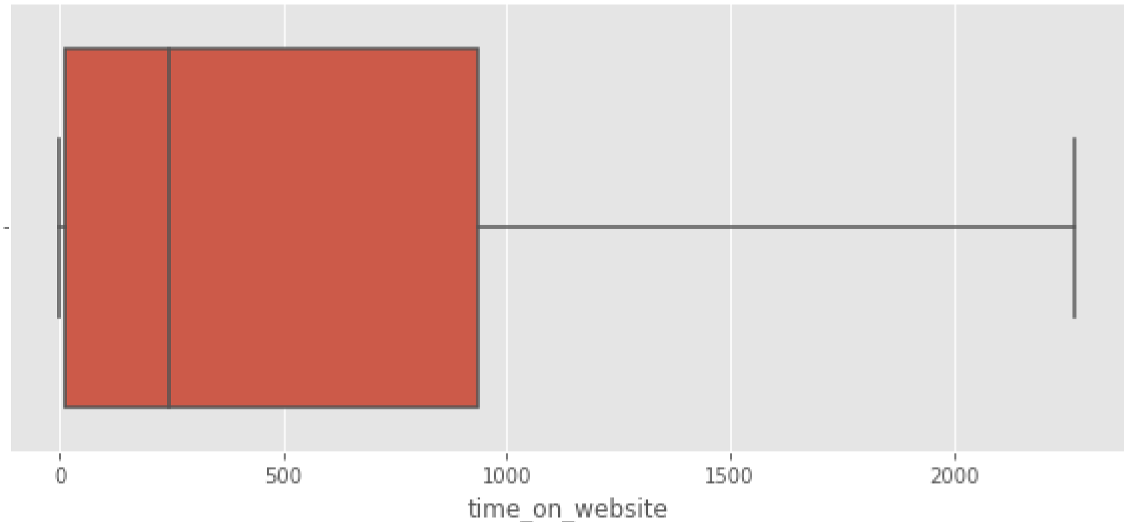


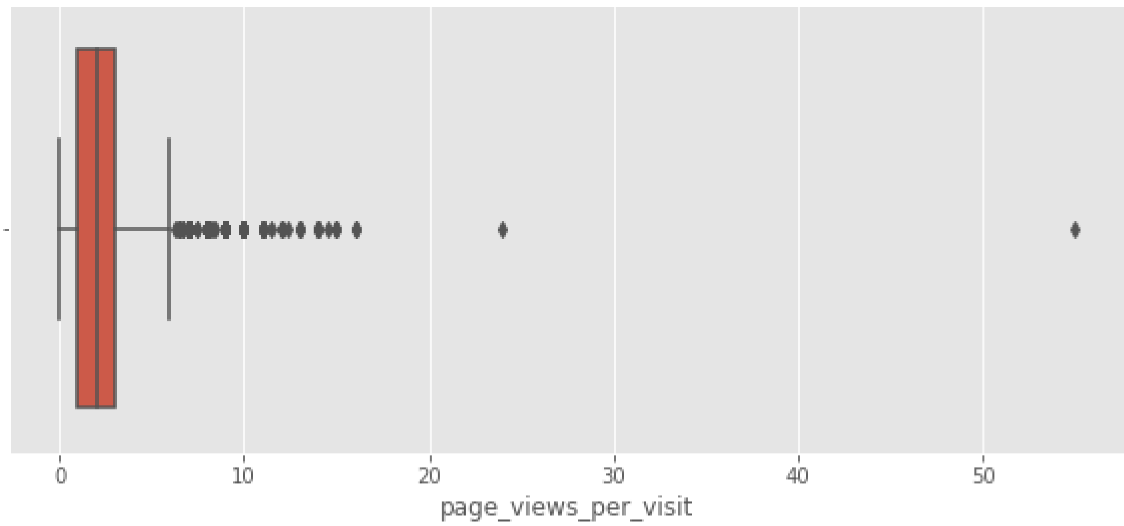
Observations: No significaqnt correlation such that columns can be dropped

Check for outliers

```
plt.figure(figsize = (10, 14))

plt.subplot(3,1,1)
sns.boxplot(df.total_visits)

plt.subplot(3,1,2)
sns.boxplot(df.time_on_website)

plt.subplot(3,1,3)
sns.boxplot(df.page_views_per_visit)
plt.show()
```







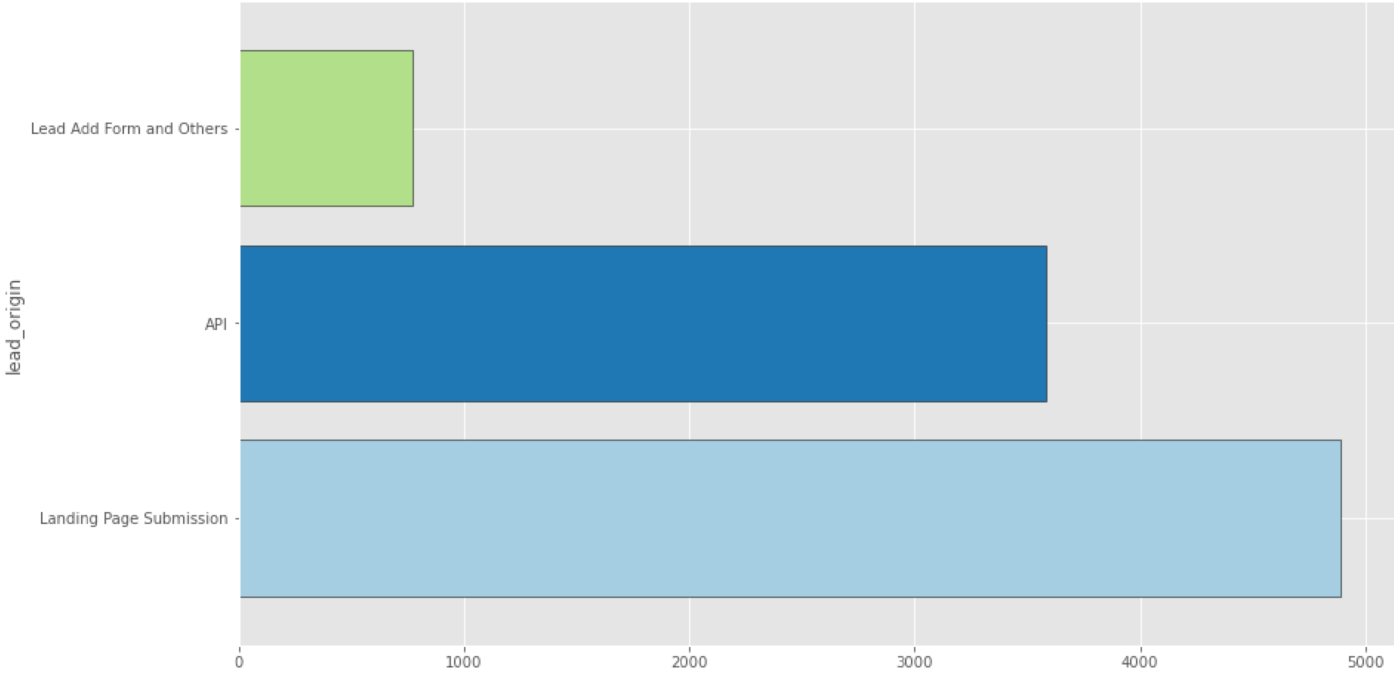
Observations

- Looking at both the box plots and the statistics, there are upper bound outliers in both `total_visits` and `page_views_per_visit` columns. We can also see that the data can be capped at 99 percentile.

Categorical columns

Lead Origin

```python
plt.figure(figsize = (14, 8))

df.groupby('lead_origin')['lead_number'].count().sort_values(ascending = False).plot(ki
                                                                                   edgecolor = 'black',
                                                                                   color = plt.cm.Paired(np.ar
plt.show()
```

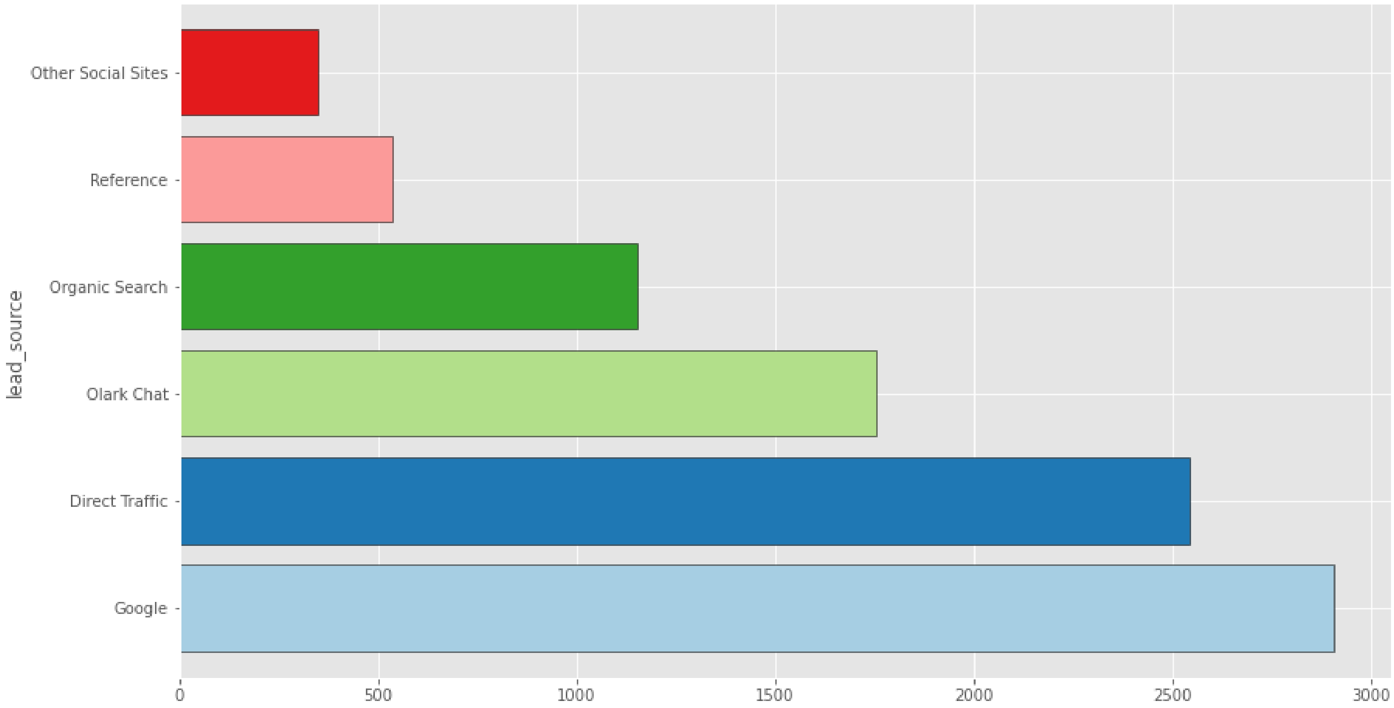



```python
df.head(1)
```

	lead_number	lead_origin	lead_source	do_not_email	converted	total_visits	time_on_website	page_views_per_visit
0	660737	API	Olark Chat	No	0	0.00	0	0.00

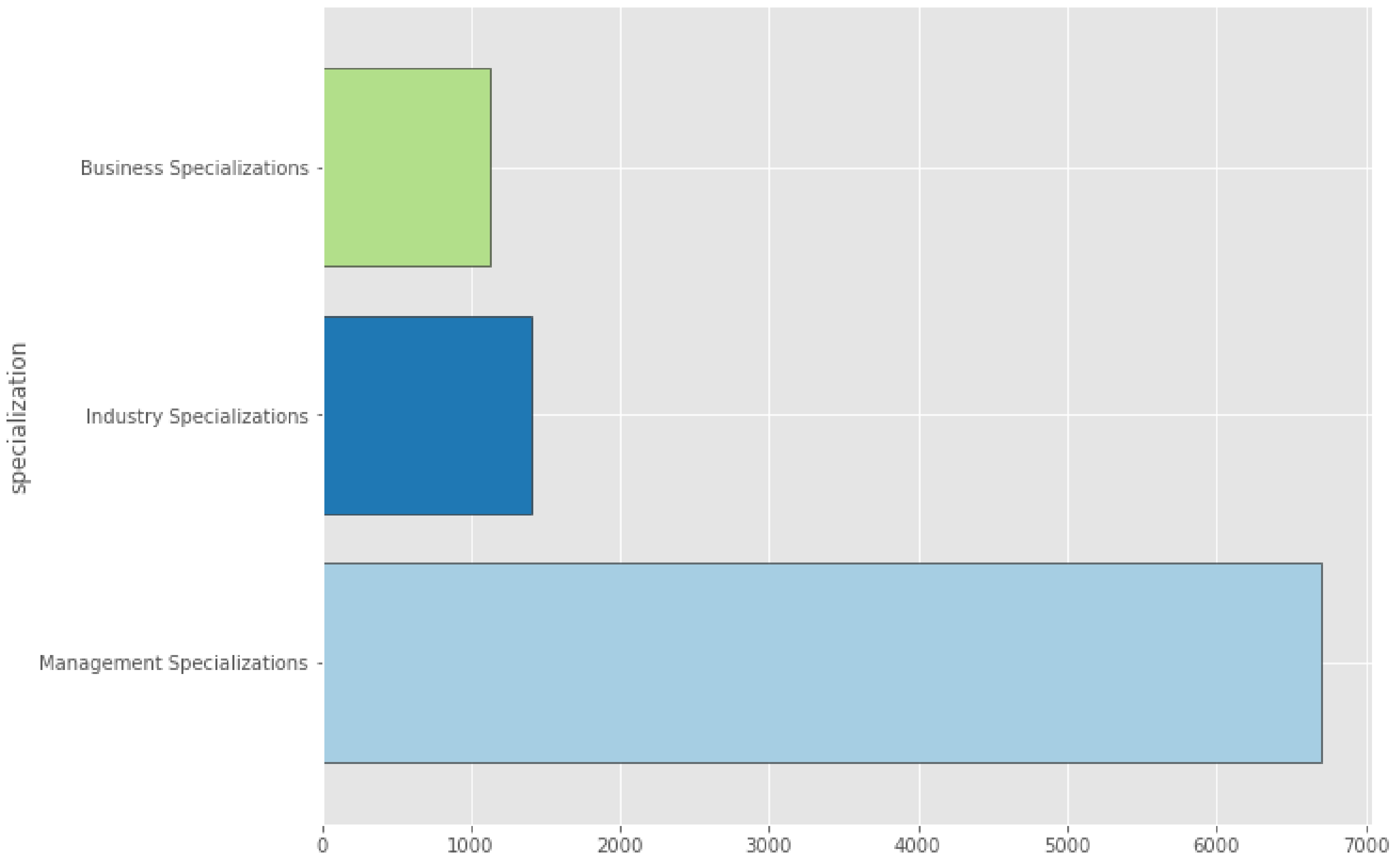
Lead Source

```python
plt.figure(figsize = (14, 8))

df.groupby('lead_source')['lead_number'].count().sort_values(ascending = False).plot(ki
                                                                                   edgecolor = 'black',
                                                                                   color = plt.cm.Paired(np.ar
plt.show()
```



Specialization

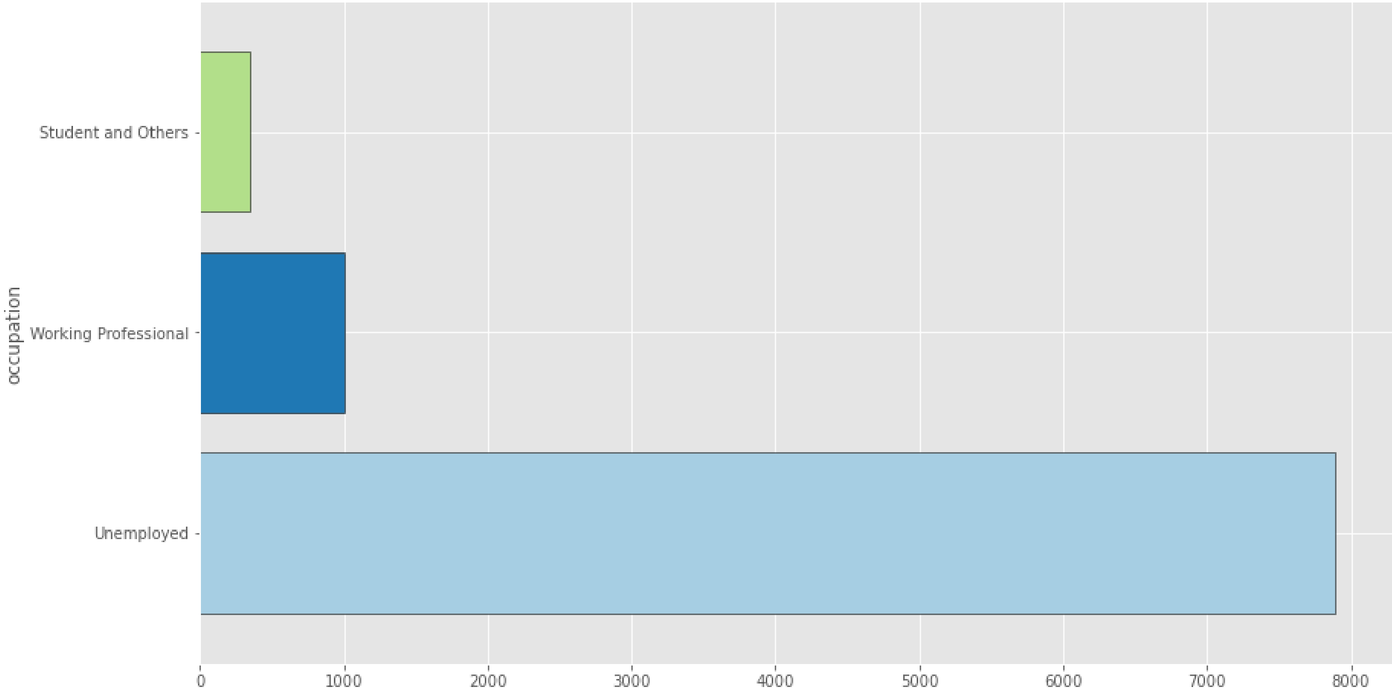
```
plt.figure(figsize = (10, 8))

df.groupby('specialization')['lead_number'].count().sort_values(ascending = False).plot
                                                                  edgecolor = 'black',
                                                                  color = plt.cm.Paired(np.ar
plt.show()
```



Most of the speciliazation taken are management

Occupation

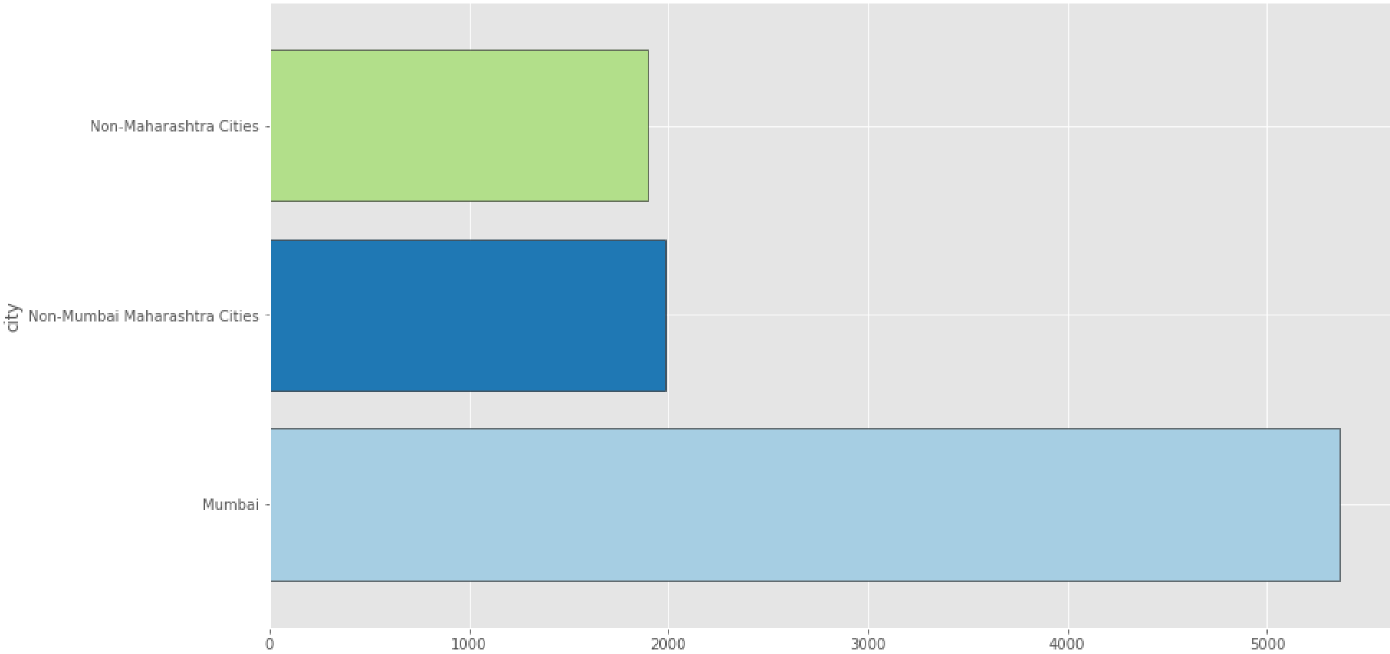
```
plt.figure(figsize = (14, 8))

df.groupby('occupation')['lead_number'].count().sort_values(ascending = False).plot(kir
                                                            edgecolor = 'black',
                                                            color = plt.cm.Paired(np.ar
plt.show()
```



Unempployed users are the most significant leads

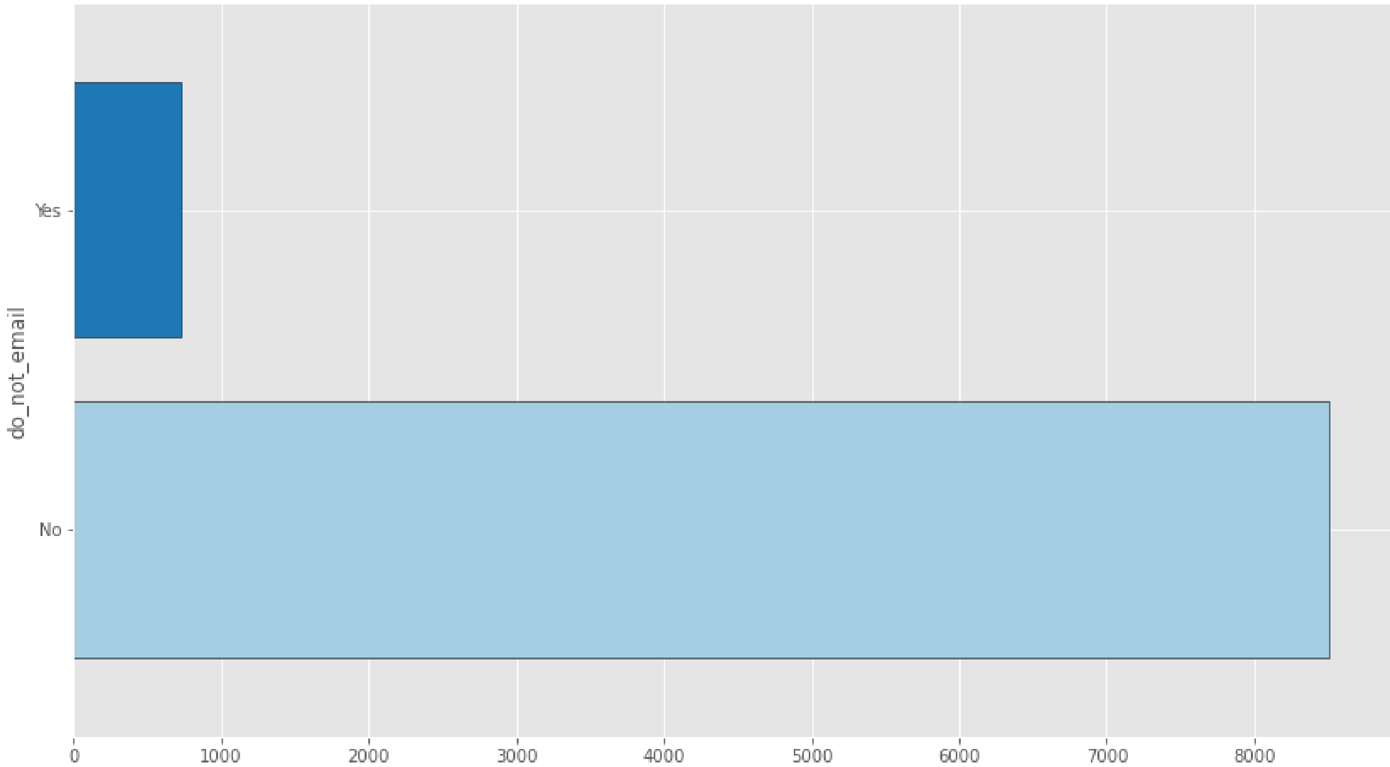
City

```
plt.figure(figsize = (14, 8))

df.groupby('city')['lead_number'].count().sort_values(ascending = False).plot(kind= 'ba
                                                            edgecolor = 'black',
                                                            color = plt.cm.Paired(np.ar
plt.show()
```



Mumbai in particular and Maharashtra in general dominates the lead. This is likely due to the fact that the courses are based in Mumbai

```
plt.figure(figsize = (14, 8))

df.groupby('do_not_email')['lead_number'].count().sort_values(ascending = False).plot(k
                                                                    edgecolor = 'black',
                                                                    color = plt.cm.Paired(np.ar
plt.show()
```



Data Preparation

Converting Binary (Yes/No) to 0/1

```
# determine unique values
for k, v in df.select_dtypes(include='object').nunique().to_dict().items():
    print('{} = {}'.format(k,v))
```

```
lead_number = 9240
lead_origin = 3
lead_source = 6
do_not_email = 2
specialization = 3
occupation = 3
city = 3
mastering_interview = 2
```

We have two binary columns: `do_not_email` , `mastering_interview`

```
binlist = ['do_not_email', 'mastering_interview']

# Defining the map function
def binary_map(x):
    return x.map({'Yes': 1, "No": 0})

# Applying the function to the housing list
df[binlist] = df[binlist].apply(binary_map)
```

```
# check the operation was success
df.head()
```

	lead_number	lead_origin	lead_source	do_not_email	converted	total_visits	time_on_website	page_views_per_visit
0	660737	API	Olark Chat	0	0	0.00	0	0.00
1	660728	API	Organic Search	0	0	5.00	674	2.50
2	660727	Landing Page Submission	Direct Traffic	0	1	2.00	1532	2.00
3	660719	Landing Page Submission	Direct Traffic	0	0	1.00	305	1.00
4	660681	Landing Page Submission	Google	0	1	2.00	1428	1.00

Creating dummy variable for categorical columns

Categorical columns are: `lead_origin` , `lead_source` , `specialization` , `occupation` , `city`

```
# Creating a dummy variable for some of the categorical variables and dropping the firs
dummy1 = pd.get_dummies(df[['lead_origin', 'lead_source', 'specialization', 'occupation

# Adding the results to the master dataframe
df = pd.concat([df, dummy1], axis=1)
```

```
# Dropping the columns for which dummies have been created
df.drop(['lead_origin', 'lead_source', 'specialization', 'occupation', 'city'], axis =

df.head()
```

Outliers Treatment

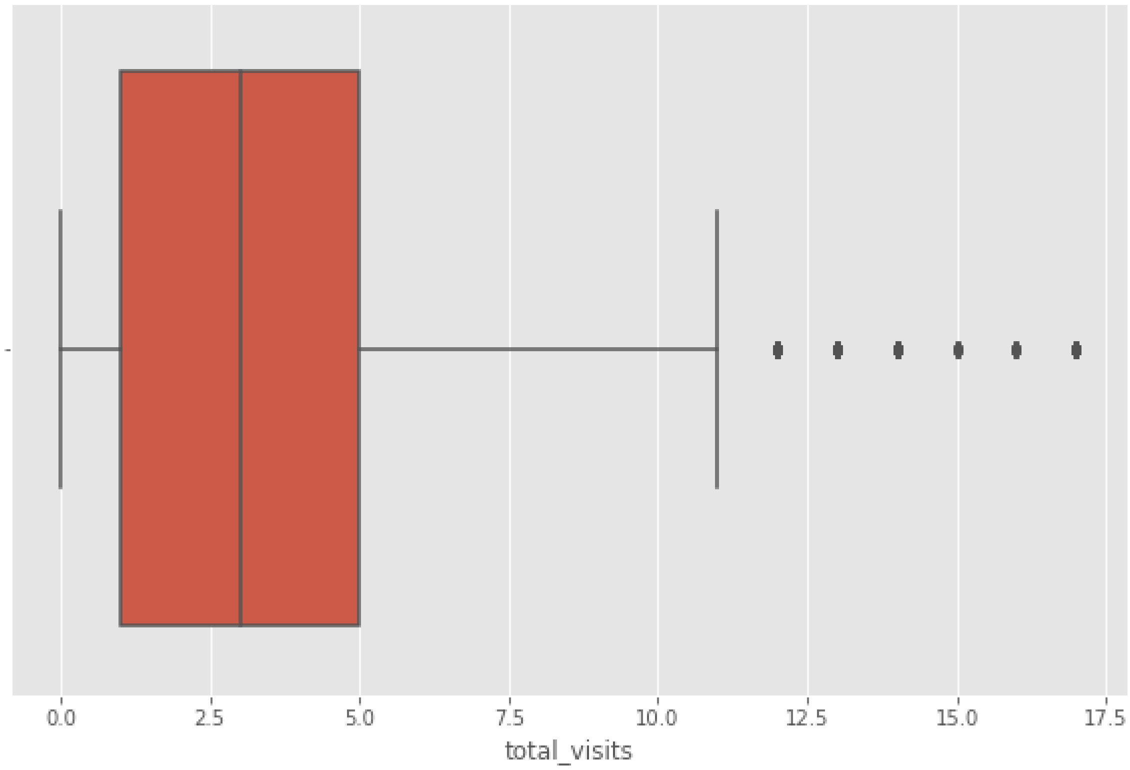
```
num_cols = df[['total_visits', 'time_on_website', 'page_views_per_visit']]

# Checking outliers at 25%, 50%, 75%, 90%, 95% and 99%
num_cols.describe(percentiles=[.25, .5, .75, .90, .95, .99])
```

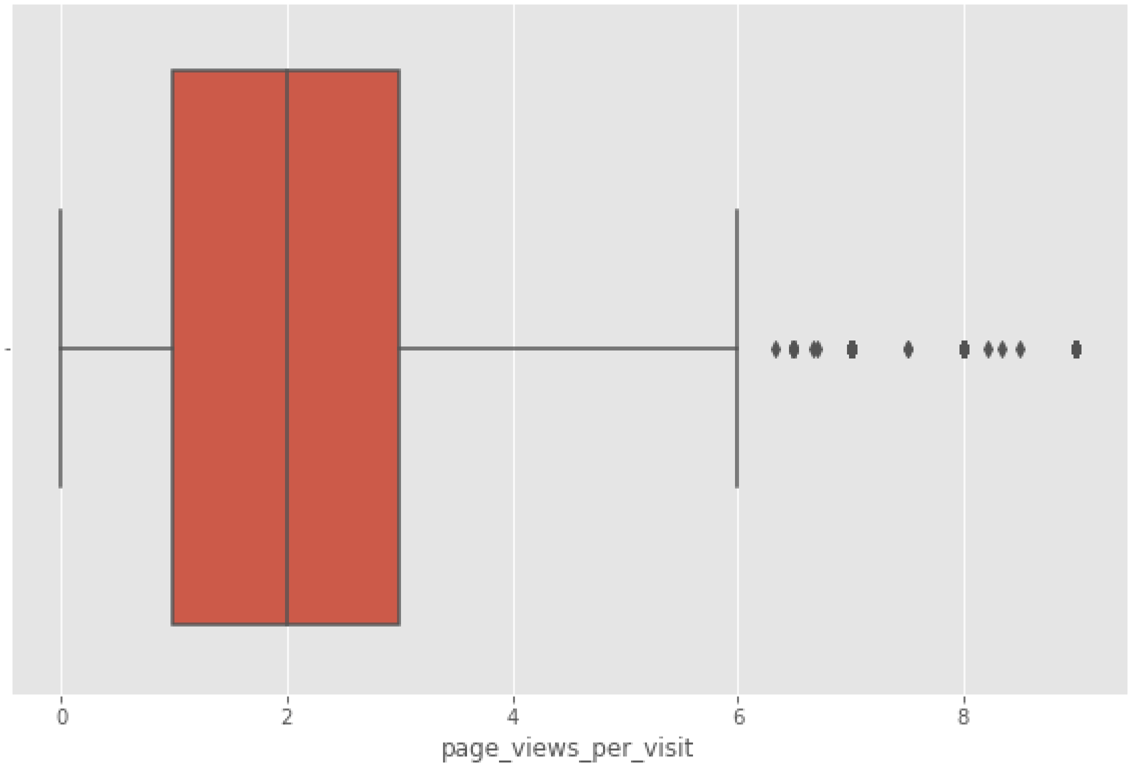
	total_visits	time_on_website	page_views_per_visit
count	9240.00	9240.00	9240.00
mean	3.44	487.70	2.36
std	4.82	548.02	2.15
min	0.00	0.00	0.00
25%	1.00	12.00	1.00
50%	3.00	248.00	2.00

	total_visits	time_on_website	page_views_per_visit
75%	5.00	936.00	3.00
90%	7.00	1380.00	5.00
95%	10.00	1562.00	6.00
99%	17.00	1840.61	9.00
max	251.00	2272.00	55.00

```
# capping at 99 percentile
df.total_visits.loc[df.total_visits >= df.total_visits.quantile(0.99)] = df.total_visit
df.page_views_per_visit.loc[df.page_views_per_visit >=
                            df.page_views_per_visit.quantile(0.99)] = df.page_views_per
```

```
plt.figure(figsize = (10, 14))

plt.subplot(2,1,1)
sns.boxplot(df.total_visits)

plt.subplot(2,1,2)
sns.boxplot(df.page_views_per_visit)
plt.show()
```





As we can see, we were able to significantly reduce the number of outliers by capping

Test-Train Split

```
# Putting feature variable to X
X = df.drop(['lead_number', 'converted'], axis=1)

X.head(1)
```

	do_not_email	total_visits	time_on_website	page_views_per_visit	mastering_interview	lead_origin_Landing Page Submission	lead_origir Add Foi
0	0	0.00	0	0.00	0	0	

```
# Putting response variable to y
y = df['converted']

y.head(1)
```

```
0    0
Name: converted, dtype: int64
```

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3
```

Feature Scaling

```
scaler = StandardScaler()

X_train[['total_visits','time_on_website','page_views_per_visit']] = scaler.fit_transfo
    X_train[['total_visits','time_on_website','page_views_per_visit']])

X_train.head()
```

	do_not_email	total_visits	time_on_website	page_views_per_visit	mastering_interview	lead_origin_Landing Page Submission	lead_o Add
1871	0	-1.02	-0.89	-1.18	0	0	
6795	0	0.21	0.01	-0.50	1	1	
3516	0	0.51	-0.69	0.09	0	0	
8105	0	0.51	1.37	1.36	0	1	
3934	0	-1.02	-0.89	-1.18	0	0	

```
# checking the conversion rate
conversion = (sum(df['converted'])/len(df['converted'].index))*100
conversion
```

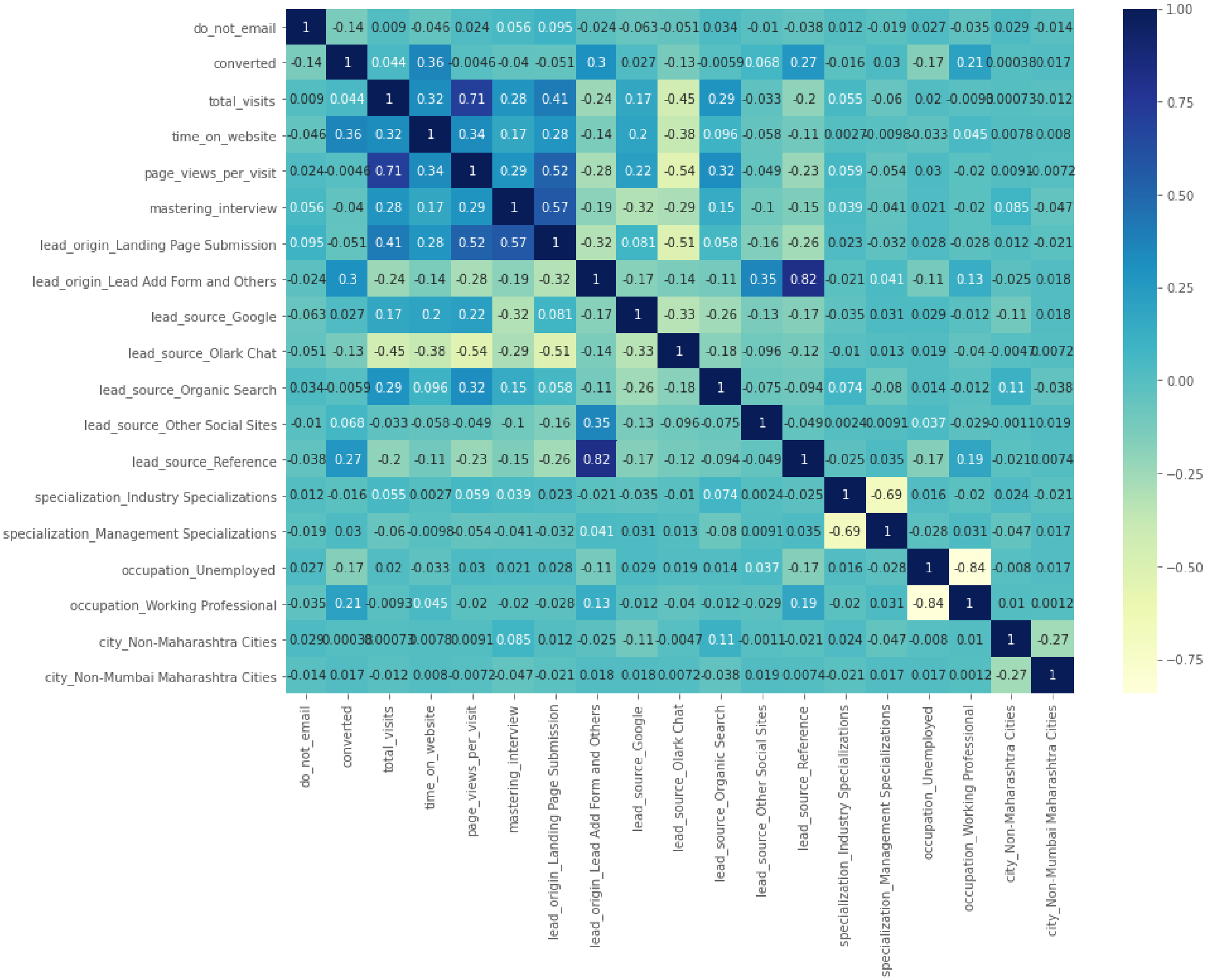
```
38.53896103896104
```

The conversion rate is 38.5%

Looking at correlations

```
# Let's see the correlation matrix
plt.figure(figsize = (14,10))
```

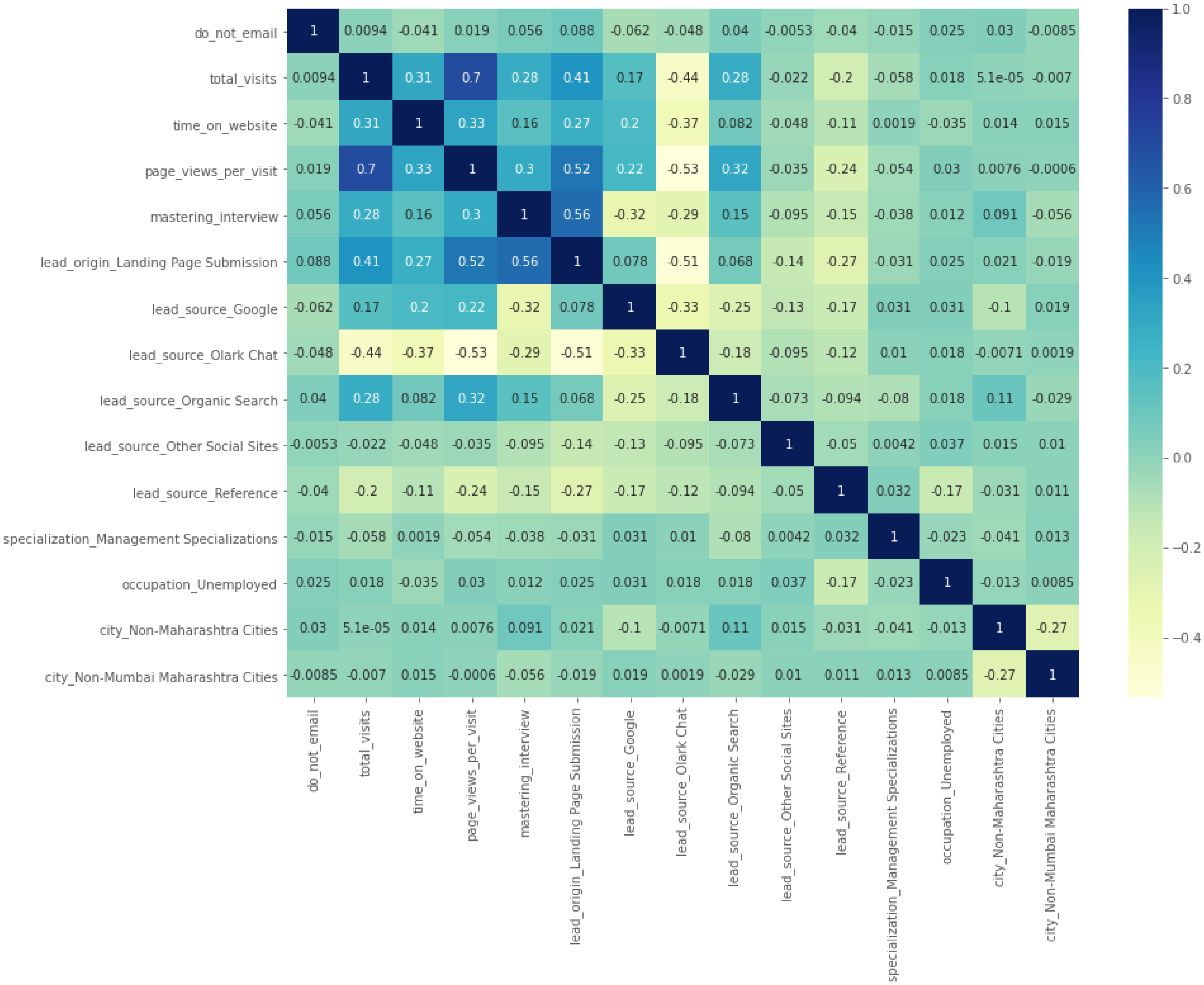
```
sns.heatmap(df.corr(),annot = True, cmap="YlGnBu")
plt.show()
```



Drop highly correlated dummy variables

```
X_test.drop(['lead_origin_Lead Add Form and Others', 'specialization_Industry Specializ
                    'occupation_Working Professional'], axis = 1, inplace = True)

X_train.drop(['lead_origin_Lead Add Form and Others', 'specialization_Industry Speciali
                    'occupation_Working Professional'], axis = 1, inplace = True)
```

```
## lets check the correlation matrix again
plt.figure(figsize = (14,10))
sns.heatmap(X_train.corr(),annot = True, cmap="YlGnBu")
plt.show()
```



Model Building

Model 1: All variables

```
# Logistic regression model
logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Binomial())
logm1.fit().summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	converted	**No. Observations:**	6468
Model:	GLM	**Df Residuals:**	6452
Model Family:	Binomial	**Df Model:**	15
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3304.7
Date:	Sun, 06 Dec 2020	**Deviance:**	6609.4
Time:	05:01:08	**Pearson chi2:**	6.64e+03
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.4513	0.147	-3.080	0.002	-0.739	-0.164
do_not_email	-1.2167	0.144	-8.467	0.000	-1.498	-0.935
total_visits	0.1403	0.042	3.356	0.001	0.058	0.222
time_on_website	1.0456	0.036	29.388	0.000	0.976	1.115
page_views_per_visit	-0.1787	0.048	-3.696	0.000	-0.273	-0.084
mastering_interview	-0.0088	0.094	-0.094	0.925	-0.192	0.175
lead_origin_Landing Page Submission	-0.0012	0.092	-0.013	0.990	-0.182	0.180
lead_source_Google	0.3639	0.100	3.625	0.000	0.167	0.561
lead_source_Olark Chat	0.6743	0.137	4.935	0.000	0.406	0.942
lead_source_Organic Search	0.2207	0.116	1.903	0.057	-0.007	0.448
lead_source_Other Social Sites	1.6136	0.175	9.202	0.000	1.270	1.957
lead_source_Reference	3.9674	0.221	17.945	0.000	3.534	4.401
specialization_Management Specializations	0.1367	0.069	1.974	0.048	0.001	0.272
occupation_Unemployed	-0.8398	0.085	-9.889	0.000	-1.006	-0.673
city_Non-Maharashtra Cities	0.1526	0.078	1.948	0.051	-0.001	0.306
city_Non-Mumbai Maharashtra Cities	0.0745	0.077	0.972	0.331	-0.076	0.225

Feature selection using RFE

```
# initiate logistic regression
logreg = LogisticRegression()

# initiate rfe
rfe = RFE(logreg, 13)             # running RFE with 13 variables as output
rfe = rfe.fit(X_train, y_train)
```

```
rfe.support_
```

```
array([ True,  True,  True,  True, False, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True])
```

```
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

```
[('do_not_email', True, 1),
 ('total_visits', True, 1),
 ('time_on_website', True, 1),
 ('page_views_per_visit', True, 1),
 ('mastering_interview', False, 2),
 ('lead_origin_Landing Page Submission', False, 3),
 ('lead_source_Google', True, 1),
 ('lead_source_Olark Chat', True, 1),
 ('lead_source_Organic Search', True, 1),
 ('lead_source_Other Social Sites', True, 1),
 ('lead_source_Reference', True, 1),
```

```
('specialization_Management Specializations', True, 1),
('occupation_Unemployed', True, 1),
('city_Non-Maharashtra Cities', True, 1),
('city_Non-Mumbai Maharashtra Cities', True, 1)]
```

```
# assign columns
col = X_train.columns[rfe.support_]
```

```
# check what columns were not selected by RFE
X_train.columns[~rfe.support_]
```

```
Index(['mastering_interview', 'lead_origin_Landing Page Submission'], dtype='object')
```

Model 2: Assessing the model with statsmodel

```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	converted	**No. Observations:**	6468
Model:	GLM	**Df Residuals:**	6454
Model Family:	Binomial	**Df Model:**	13
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3304.7
Date:	Sun, 06 Dec 2020	**Deviance:**	6609.4
Time:	05:08:43	**Pearson chi2:**	6.64e+03
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-0.4592	0.112	-4.082	0.000	-0.680	-0.239
do_not_email	-1.2163	0.144	-8.474	0.000	-1.498	-0.935
total_visits	0.1399	0.042	3.364	0.001	0.058	0.221
time_on_website	1.0456	0.036	29.393	0.000	0.976	1.115
page_views_per_visit	-0.1791	0.047	-3.811	0.000	-0.271	-0.087
lead_source_Google	0.3702	0.079	4.675	0.000	0.215	0.525
lead_source_Olark Chat	0.6807	0.113	6.032	0.000	0.460	0.902
lead_source_Organic Search	0.2240	0.108	2.068	0.039	0.012	0.436
lead_source_Other Social Sites	1.6203	0.158	10.283	0.000	1.311	1.929
lead_source_Reference	3.9740	0.207	19.210	0.000	3.569	4.379
specialization_Management Specializations	0.1368	0.069	1.975	0.048	0.001	0.273
occupation_Unemployed	-0.8397	0.085	-9.889	0.000	-1.006	-0.673

city_Non-Maharashtra Cities	0.1524	0.078	1.946	0.052	-0.001	0.306
city_Non-Mumbai Maharashtra Cities	0.0749	0.077	0.979	0.328	-0.075	0.225

Lets create empty lists of categorical columns and numerical columns, then we will review the columns one by one and see what needs to be done with each of them

```
cat= []
num = []
```

```
#Creating a function to get the column details
def details(x):
    print(df[x].value_counts())
    print(df[x].isnull().sum(),'null values')
    print(df[x].isnull().sum()/df.count()['lead_number']*100,'% values are null')
```

```
# Do not email column
details('Do Not Email')
```

we'll leave this column as is for now, and add this to a new list of binary categorical variables

```
bi_cat = []
bi_cat.append('Do Not Email')
```

```
#Do Not Call column
details('Do Not Call')
```

```
#Lets drop the column since its only two records and it doesn't make sense to keep this
df.drop('Do Not Call',axis=1,inplace = True)
#df[['Do Not Call','Converted']].value_counts()
```

```
details('TotalVisits')
```

```
# We can see that there is a lot of outliers here, we can also plot a boxplot to get a
sns.boxplot(df['TotalVisits'])
#and see the median(sometimes we impute the null values with median), and 95th to 99th
df.TotalVisits.quantile([0.50,0.95,0.96,0.97,0.98,0.99])
```

Considering the values and outliers here, Let's cap the values at 96th percentile i.e. 10

Also, we will impute the null values with 0 here, we could impute this with median but considering the values might not have been tracked because they haven't logged on to the site, and 0 is also the mode of the column.

```
df[df['TotalVisits'].isnull()]['TotalVisits'] = df['TotalVisits'].mode()[0]
#df[df.TotalVisits > df.TotalVisits.quantile(0.96)] = df.TotalVisits.quantile(0.96)
#Also add this column to our numerical columns list
num.append('TotalVisits')
```

Let's also create a function to cap the outliers if needed in future

```
def cap(col,typ='right',value=0.95):
    if typ == 'left':
        df[df[col]<df[col].quantile(value)][col] = df[col].quantile(value)
    else:
        df[df[col]>df[col].quantile(value)][col] = df[col].quantile(value)
```

```
# and capping the column as mentioned earlier
cap('TotalVisits')
```

```
# Lets look at Total Time Spent on Website column
details('Total Time Spent on Website')
```

```
###############################################################################
##Also add this column to our numerical columns list
#num.append('Total Time Spent on Website')
```

Since we also look at the boxplots and percentiles for numberical numbers, lets create a similar details function to include these two pieces of information.

```
def num_details(x):
    print(df[x].value_counts())
    print(df[x].isnull().sum(),'null values')
    print(df[x].isnull().sum()/df.count()[x]*100,'% values are null')
    print('Percentiles are as follows')
    print(df[x].quantile([0.50,0.95,0.96,0.97,0.98,0.99]))
    sns.boxplot(df[x])
```

```
#Lets look at column Page Views Per Visit
num_details('Page Views Per Visit')
```

```
df[df['Page Views Per Visit'].isnull()]['Page Views Per Visit'] = df['Page Views Per Vi
cap('Page Views Per Visit')
#Also add this column to our numerical columns list
num.append('Page Views Per Visit')
```

```
details('Last Activity')
```

```
################################################
##Also add this column to our numerical columns list
#num.append('Last Activity')
```

```
#Lets have a look at the column 'Country'
details('Country')
```

There are 27 percent null values in this column, and other values are mostly India, so this column would not be that useful, we could create a binary column using this like 'India' if there weren't so many null values here. but considering the null values, lets drop this column

```
df.drop('Country', axis =1, inplace = True)
```

```
details('Specialization')
```

```
#####################################################
```

```
details('How did you hear about X Education')
```

There are too many null values in this as well, and the most values are also not selected I think, because the option says default value 'Select', lets drop this column as well

```
df.drop('How did you hear about X Education', axis =1, inplace = True)
```

```
details('What is your current occupation')
```

```
###################################
```

```
details('What matters most to you in choosing a course')
```

```
################################
```

```
details('Search')
```

```
details('Magazine')
```

```
details('Newspaper Article')
```

```
details('X Education Forums')
```

```
details('Newspaper')
```

```
details('Digital Advertisement')
```

All these columns have high imbalance, and mostly have only one value i.e. No. So it doesn't make sense to keep these column, Lets delete these column

```
#Lets look at the column 'Through Recommendation'
details('Through Recommendations')
```

```
details('Receive More Updates About Our Courses')
```

```
df.drop(['Search','Magazine','Newspaper Article','X Education Forums','Newspaper','Digi
```

```
details('Tags')
```

```
##################
```

```
details('Lead Quality')
```

```
######################
```

```
details('Update me on Supply Chain Content')
```

```
# There's only singular value in this, so lets drop this column
df.drop('Update me on Supply Chain Content', axis=1, inplace = True)
```

```
details('Get updates on DM Content')
```

```
# There's only singular value in this, so lets drop this column
df.drop('Get updates on DM Content', axis=1, inplace = True)
```

```
details('Lead Profile')
```

```
##########################
```

```
details('City')
```

```
###################
```

```
details('Asymmetrique Activity Index')
```

```
#################
```

```
details('Asymmetrique Profile Index')
```

```
###################
```

```
details('Asymmetrique Activity Score')
```

```
########################
```

```
details('Asymmetrique Profile Score')
```

```
#################
```

```
details('I agree to pay the amount through cheque')
```

```
# There's only singular value in this, so lets drop this column
df.drop('I agree to pay the amount through cheque', axis=1, inplace = True)
```

```
details('A free copy of Mastering The Interview')
```

we'll leave this column as is for now, and add this to a new list of binary categorical variables

```
bi_cat.append('A free copy of Mastering The Interview')
```

```
details('Last Notable Activity')
```

```
##############################
```

```
#Lets have a look at our three categories of column
print(cat)
```

```python
print(num)
print(bi_cat)
```

```python
df.head(1)
```

```python

```