Story behind..

There has been a revenue decline for the bank and they would like to know what actions to take. After investigation, it was found that the root cause is that their clients are not depositing as frequently as before. Term deposits allow banks to hold onto a deposit for a specific amount of time, so banks can lend more and thus make more profits. In addition, banks also hold better chance to persuade term deposit clients into buying other products such as funds or insurance to further increase their revenues.

Variables:

idUnique -identifier for each sample in the dataset. Cannot be used for modelling
customer_age -Age of the Customer in years
job_type - Type of job of the customer
marital - Marital Status of the Custmer
education - Education Level of the Customer
default Whether - customer has Defaulted in Past
balance Current - Balance in the Customer's Bank
housing_loan - Has customer taken a Housing Loan
personal_loan - Has customer taken a Personal Loan
communication_type -Type of communication made by the bank with the customer
day_of_month - Day of month of the last contact made with customer
month -Month for the last contact made with customer
last_contact_duration -Last Contact duration made with the customer (in seconds)
num_contacts_in_campaign -Number of contacts made with the customer during the current campaign.
days_since_prev_campaign_contact - Number of days passed since customer was contacted in previous campaign.
num_contacts_prev_campaign -Number of contacts made with the customer during the previous campaign.
prev_campaign_outcome -Success or Failure in previous Campaign.
term_deposit_subscribed (Target) - Has the customer taken a term deposit ?

Agenda:

```
Brief look at data
Data shape
Traget distribution
Variable datatypes
Null values
Unique values
Separating categorical ans numerical columns
```

Impoort All Required Labraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [28]:
```
train = pd.read_csv('Train_data.csv')
test = pd.read_csv('Test_1.csv')
train
```

Out[28]:

	id	customer_age	job_type	marital	education	default	balance	housing_loan
0	id_43823	28.0	management	single	tertiary	no	285.0	yes
1	id_32289	34.0	blue-collar	married	secondary	no	934.0	no
2	id_10523	46.0	technician	married	secondary	no	656.0	no
3	id_43951	34.0	services	single	secondary	no	2.0	yes
4	id_40992	41.0	blue-collar	married	primary	no	1352.0	yes
...	...	...	...	...	...	...	...	...
31642	id_27290	58.0	admin.	married	secondary	no	567.0	yes
31643	id_20428	51.0	management	married	tertiary	no	1072.0	no
31644	id_44679	41.0	unemployed	married	primary	no	242.0	yes
31645	id_4841	48.0	services	married	secondary	no	2699.0	no
31646	id_1723	38.0	technician	single	tertiary	no	1045.0	no

31647 rows × 18 columns

Brief look at data

In [3]:
```
train.head()
```

Out[3]:

	id	customer_age	job_type	marital	education	default	balance	housing_loan	per
0	id_43823	28.0	management	single	tertiary	no	285.0	yes	
1	id_32289	34.0	blue-collar	married	secondary	no	934.0	no	
2	id_10523	46.0	technician	married	secondary	no	656.0	no	
3	id_43951	34.0	services	single	secondary	no	2.0	yes	
4	id_40992	41.0	blue-collar	married	primary	no	1352.0	yes	

In [4]:
```
test.head()
```

Out[4]:

	id	customer_age	job_type	marital	education	default	balance	housing_loan	pers
0	id_17231	55.0	retired	married	tertiary	no	7136.0	no	
1	id_34508	24.0	blue-collar	single	secondary	no	179.0	yes	
2	id_44504	46.0	technician	divorced	secondary	no	143.0	no	
3	id_174	56.0	housemaid	single	unknown	no	6023.0	no	
4	id_2115	62.0	retired	married	secondary	no	2913.0	no	

Dataset shape

In [5]:
```
id_col, target_col = 'id', 'term_deposit_subscribed'
```

In [6]:
```
print('Train contains',train.shape[0],'samples and ',train.shape[1],'variables
print('Test contains',test.shape[0],'samples and ',test.shape[1],'variables')

features = [c for c in train.columns if c not in [id_col, target_col]]
print('There are',len(features),'number of features')
```

```
Train contains 31647 samples and  18 variables
Test contains 13564 samples and  17 variables
There are 16 number of features
```

Traget distribution

Normalize the data to get ratio instead of raw count

In [7]:
```
train[target_col].value_counts(normalize=True)
```

Out[7]:
```
term_deposit_subscribed
0    0.892754
1    0.107246
Name: proportion, dtype: float64
```

In [8]:
```
sns.countplot(train[target_col].value_counts(normalize=True))
plt.title('Target distribution')
plt.show()
```

More than 25,000 have not subscribet to the term deposit which is 89% and only 10% have subscribed.

Variable datatypes

In [9]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31647 entries, 0 to 31646
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   id                                31647 non-null  object 
 1   customer_age                      31028 non-null  float64
 2   job_type                          31647 non-null  object 
 3   marital                           31497 non-null  object 
 4   education                         31647 non-null  object 
 5   default                           31647 non-null  object 
 6   balance                           31248 non-null  float64
 7   housing_loan                      31647 non-null  object 
 8   personal_loan                     31498 non-null  object 
 9   communication_type                31647 non-null  object 
 10  day_of_month                      31647 non-null  int64  
 11  month                             31647 non-null  object 
 12  last_contact_duration             31336 non-null  float64
 13  num_contacts_in_campaign          31535 non-null  float64
 14  days_since_prev_campaign_contact  5816 non-null   float64
 15  num_contacts_prev_campaign        31647 non-null  int64  
 16  prev_campaign_outcome             31647 non-null  object 
 17  term_deposit_subscribed           31647 non-null  int64  
dtypes: float64(5), int64(3), object(10)
memory usage: 4.3+ MB
```

We have large number of categorical values and few numerical values. We will analyse them separately in later stage.

Null values

In [10]:
```
null_value_percentage = (train.isnull().sum()/train.shape[0])*100
null_value_percentage.sort_values(ascending = False)
```

Out[10]:
```
days_since_prev_campaign_contact    81.622271
customer_age                         1.955952
balance                              1.260783
last_contact_duration                0.982716
marital                              0.473979
personal_loan                        0.470819
num_contacts_in_campaign             0.353904
id                                   0.000000
month                                0.000000
prev_campaign_outcome                0.000000
num_contacts_prev_campaign           0.000000
communication_type                   0.000000
day_of_month                         0.000000
housing_loan                         0.000000
default                              0.000000
education                            0.000000
job_type                             0.000000
term_deposit_subscribed              0.000000
dtype: float64
```

days_since_prev_campaign_contact has 81% missing data. The reason might be that these customers were never reached during previous campaign. Remaining variables habe very small percentage of missing values which will not matter much.

Unique values

In [11]:
```
train.nunique()
```

Out[11]:
```
id                                  31647
customer_age                           77
job_type                               12
marital                                 3
education                               4
default                                 2
balance                              6563
housing_loan                            2
personal_loan                           2
communication_type                      3
day_of_month                           31
month                                  12
last_contact_duration                1447
num_contacts_in_campaign               46
days_since_prev_campaign_contact      511
num_contacts_prev_campaign             41
prev_campaign_outcome                   4
term_deposit_subscribed                 2
dtype: int64
```

```
There are lot of unique values. day_of_month has 31 unique valus which is
obvious and months as 12.

Separating categorical and numerical columns
```

In [14]:
```python
#looping through the columns
#check if datatype is object('O')
#if yes add to list
cat_cols = [train.columns[i]
            for i in range(1, train.shape[1]-1)
            if train.iloc[:,i].dtype=='O']
cat_cols
```

Out[14]:
```
['job_type',
 'marital',
 'education',
 'default',
 'housing_loan',
 'personal_loan',
 'communication_type',
 'month',
 'prev_campaign_outcome']
```

In [15]:
```python
num_cols = [c for c in features if c not in cat_cols]
num_cols
```

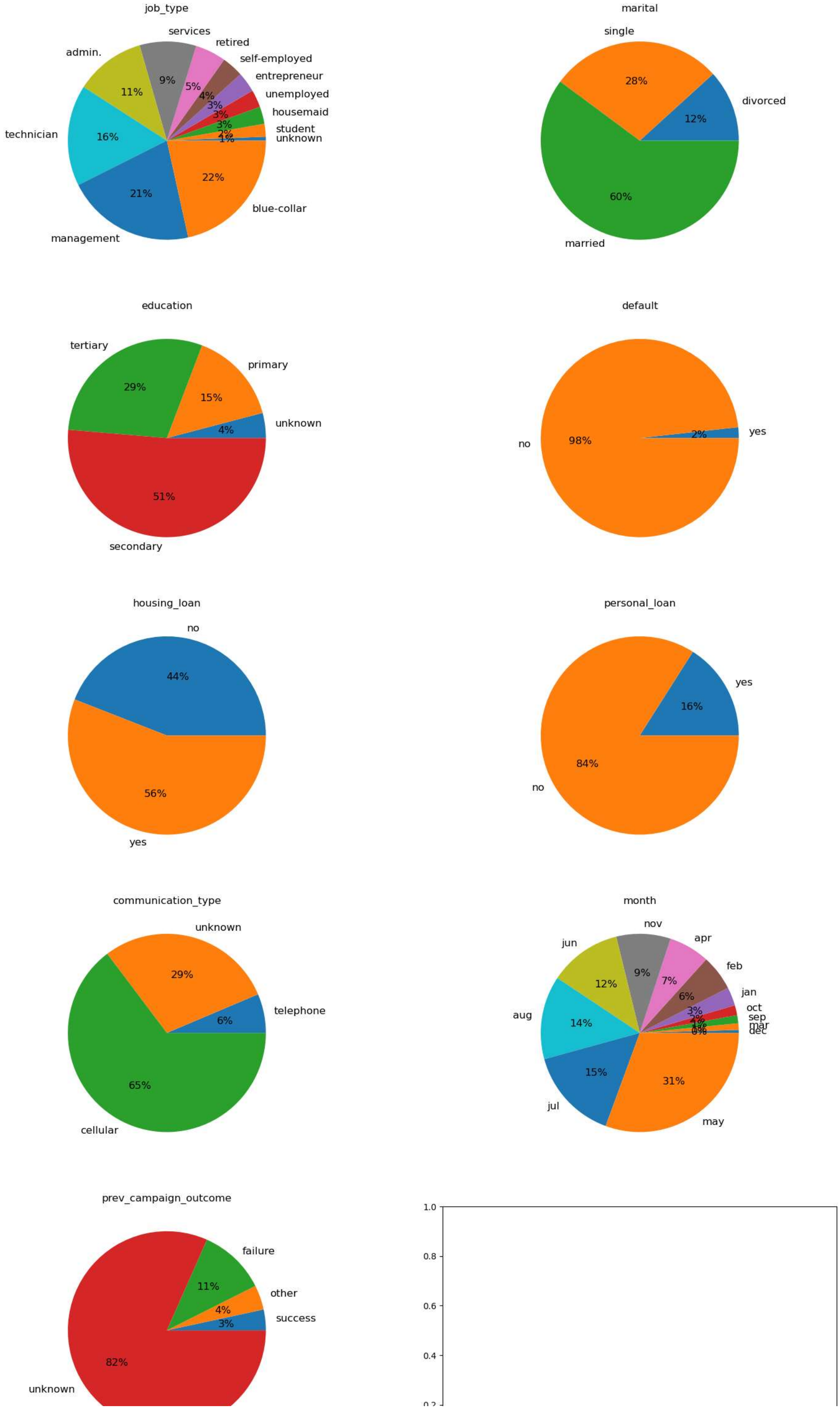
Out[15]:
```
['customer_age',
 'balance',
 'day_of_month',
 'last_contact_duration',
 'num_contacts_in_campaign',
 'days_since_prev_campaign_contact',
 'num_contacts_prev_campaign']
```

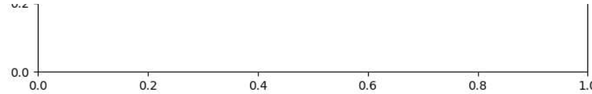
Univariate analysis of Categorical features Pick one variable one at a time and analyse individually like frequency, distribution etc.

1. Pie chart to see propotion of samples

In [17]:
```
fig, axes = plt.subplots(5, 2, figsize=(18,30))
axes = [ax for axes_rows in axes for ax in axes_rows]

for i, c in enumerate(train[cat_cols]):
    train[c].value_counts()[::-1].plot(kind='pie',
                                        ax=axes[i],
                                        title=c,
                                        autopct='%.0f%%',
                                        fontsize=12)
    axes[i].set_ylabel('')
```


job_type
services
retired
admin.
self-employed
entrepreneur
unemployed
housemaid
student
unknown
technician
management
blue-collar
9%
5%
4%
11%
3%
3%
3%
16%
1%
22%
21%
marital
single
divorced
28%
12%
60%
married
education
tertiary
primary
29%
15%
unknown
4%
51%
secondary
default
yes
no
98%
2%
housing_loan
no
44%
56%
yes
personal_loan
yes
16%
84%
no
communication_type
unknown
29%
telephone
6%
65%
cellular
month
nov
apr
jun
feb
9%
7%
12%
6%
jan
3%
oct
aug
sep
14%
mar
dec
15%
31%
jul
may
prev_campaign_outcome
failure
11%
other
4%
success
3%
82%
unknown
1.0
0.8
0.6
0.4
0.2

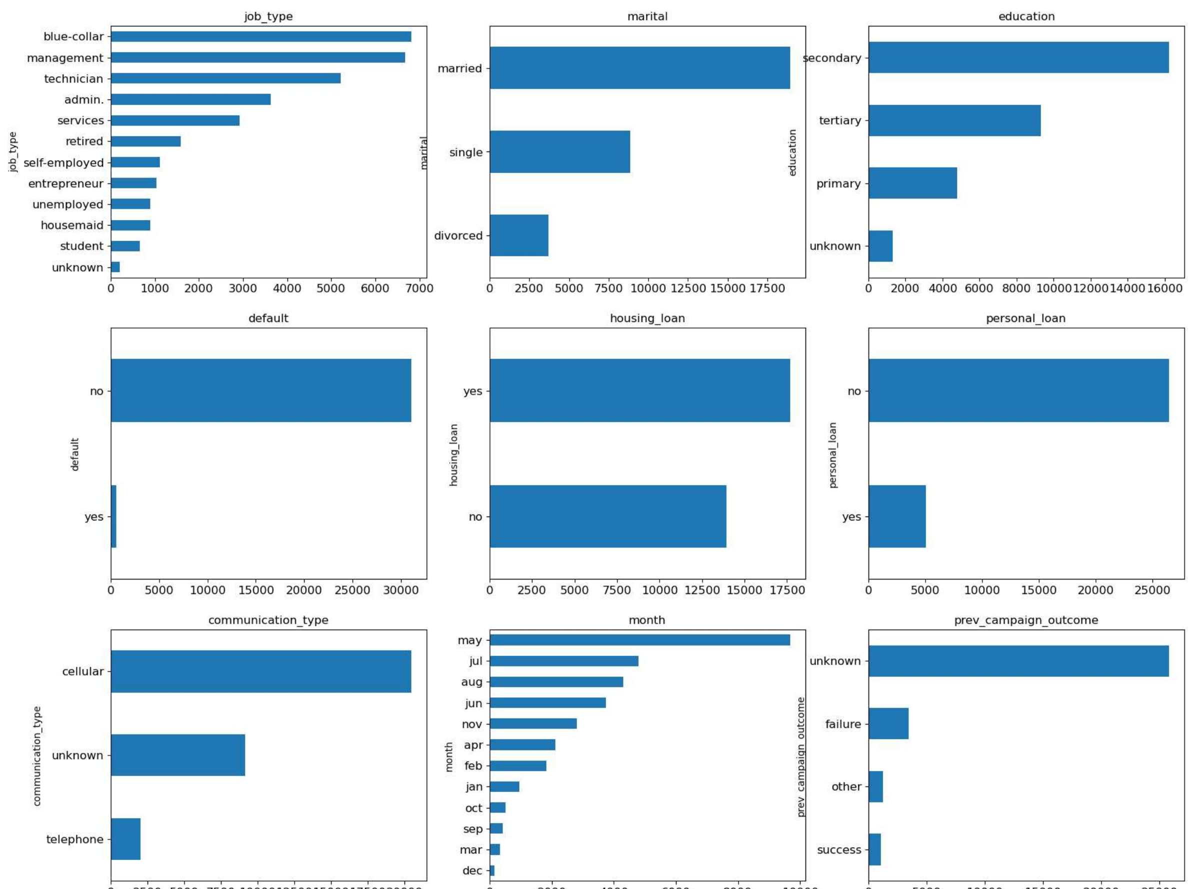


2. Bar plot to see frequency

In [18]:

```python
fig, axes = plt.subplots(3, 3, figsize=(20,16))
axes = [ax for axes_rows in axes for ax in axes_rows]

for i, c in enumerate(train[cat_cols]):
    train[c].value_counts()[::-1].plot(kind='barh',
                                       ax=axes[i],
                                       title=c,
                                       fontsize=12)
```



Observations

```
Less number of students and more number of management and technician
customers
Most of married customers
Most customers education levels is secondary
Most cutomers are not defaulted in past
More than 50% have taken housing loan
```

In [52]:
```
vc_a
```

Out[52]:

	count	proportion	term_deposit_subscribed
0	blue-collar	0.225215	0
1	management	0.206031	0
2	technician	0.166389	0
3	admin.	0.114926	0
4	services	0.095282	0
5	retired	0.044102	0
6	self-employed	0.034722	0
7	entrepreneur	0.034014	0
8	housemaid	0.029023	0
9	unemployed	0.027006	0
10	student	0.017131	0
11	unknown	0.006159	0

In [33]:
```
vc_b
```

Out[33]:

	count	proportion	term_deposit_subscribed
0	management	0.248969	1
1	technician	0.152917	1
2	blue-collar	0.133471	1
3	admin.	0.111962	1
4	retired	0.101650	1
5	services	0.068061	1
6	student	0.052740	1
7	unemployed	0.040660	1
8	self-employed	0.038303	1
9	entrepreneur	0.022392	1
10	housemaid	0.021509	1
11	unknown	0.007366	1

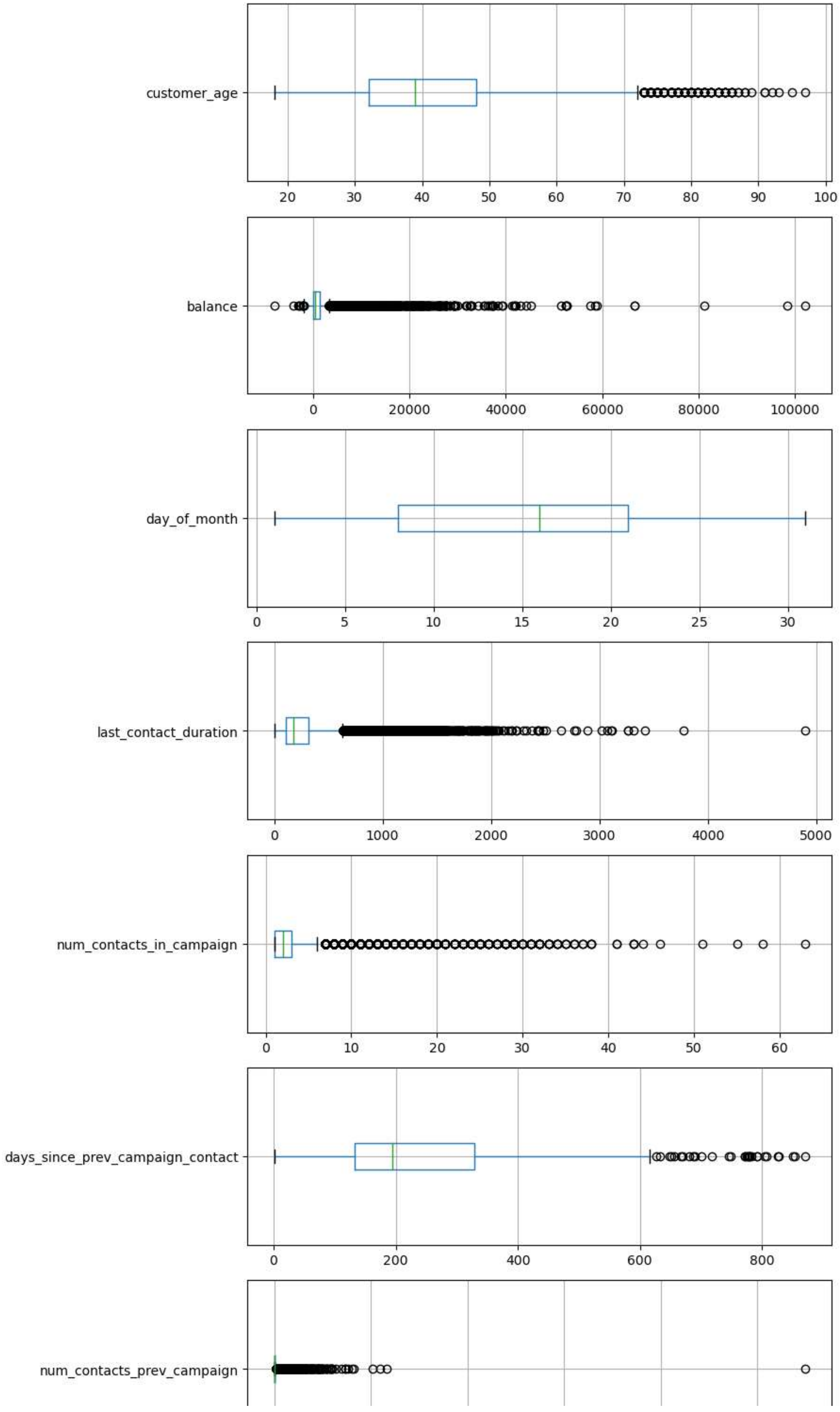
In [34]:
```
df
```

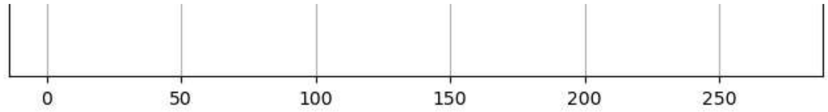
Out[34]:

	count	proportion	term_deposit_subscribed
0	blue-collar	0.225215	0
1	management	0.206031	0
2	technician	0.166389	0
3	admin.	0.114926	0
4	services	0.095282	0
5	retired	0.044102	0
6	self-employed	0.034722	0
7	entrepreneur	0.034014	0
8	housemaid	0.029023	0
9	unemployed	0.027006	0
10	student	0.017131	0
11	unknown	0.006159	0
12	management	0.248969	1
13	technician	0.152917	1
14	blue-collar	0.133471	1
15	admin.	0.111962	1
16	retired	0.101650	1
17	services	0.068061	1
18	student	0.052740	1
19	unemployed	0.040660	1
20	self-employed	0.038303	1
21	entrepreneur	0.022392	1
22	housemaid	0.021509	1
23	unknown	0.007366	1

Univariate analysis of Numerical features

In [36]:
```
fig, axes = plt.subplots(7,1,figsize=(8,20))
for i,c in enumerate(train[num_cols]):
    train[[c]].boxplot(ax=axes[i], vert=False)
```

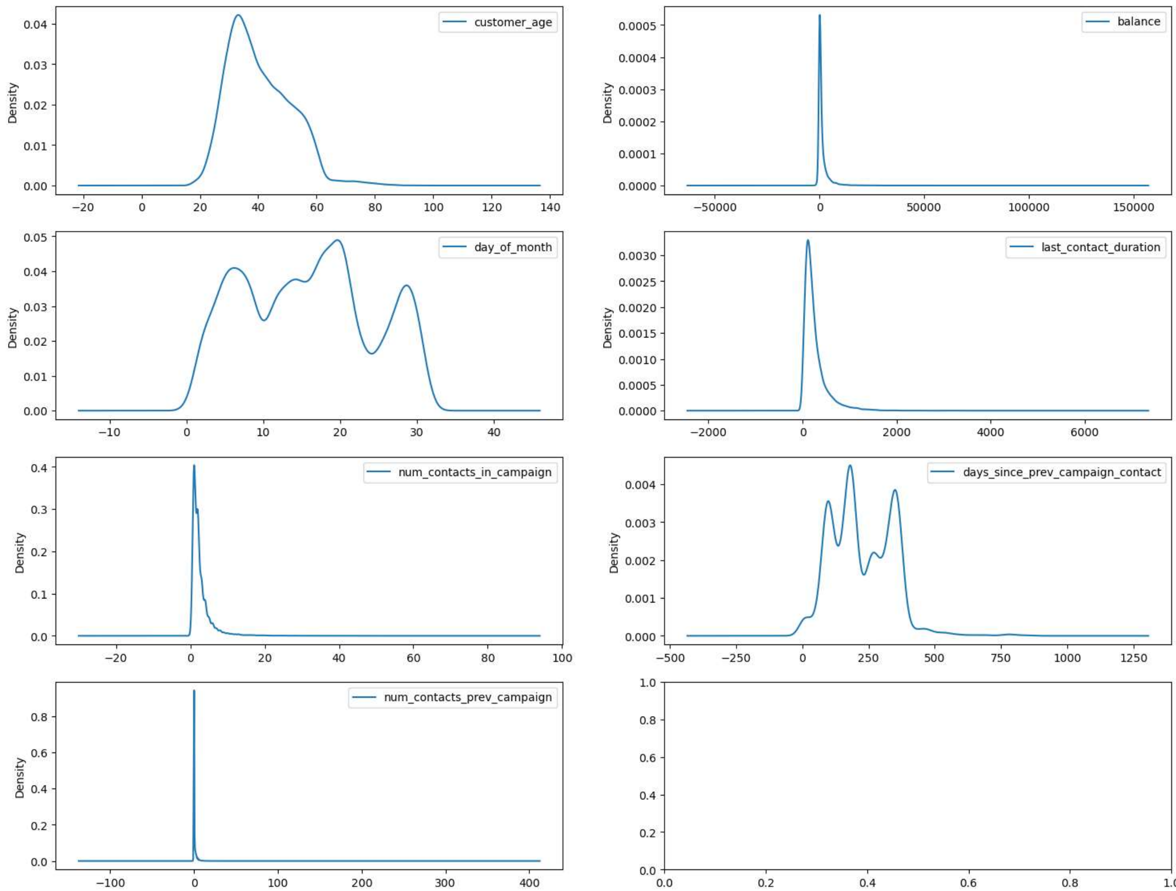

customer_age
20 30 40 50 60 70 80 90 100
balance
0 20000 40000 60000 80000 100000
day_of_month
0 5 10 15 20 25 30
last_contact_duration
0 1000 2000 3000 4000 5000
num_contacts_in_campaign
0 10 20 30 40 50 60
days_since_prev_campaign_contact
0 200 400 600 800
num_contacts_prev_campaign



In [37]:
```
#We can see many of the features have lot of outliers. Let's see distrubution

fig, axes = plt.subplots(4, 2, figsize=(18,14))
axes = [ax for axes_rows in axes for ax in axes_rows]

for i, c in enumerate(num_cols):
    plot = train[[c]].plot(kind='kde', ax=axes[i])
```



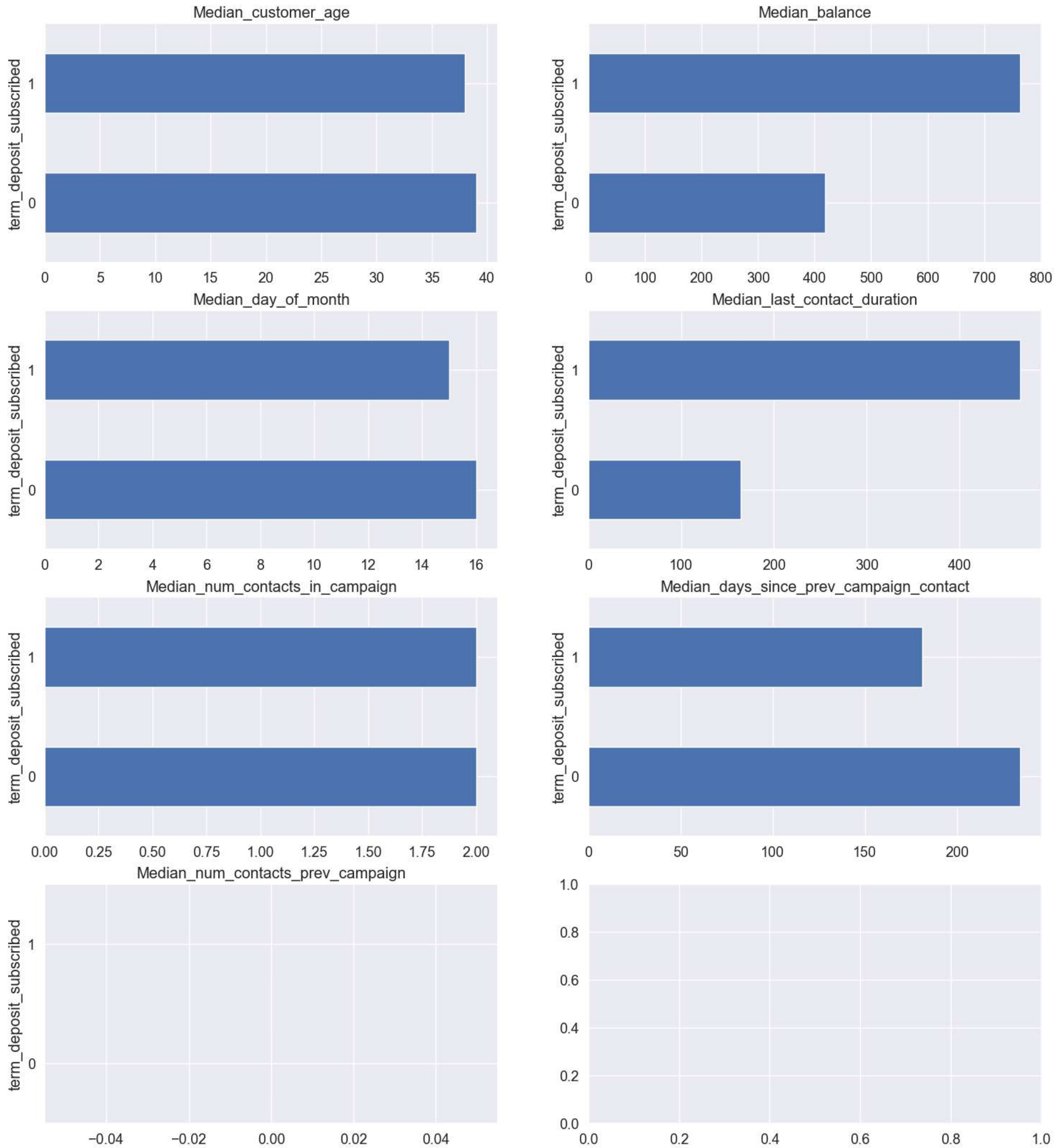

```
Observations

Most of the customers lie between age of 20 and 60
Big campaing happened roughly 180 and 365 days ago
```

Bivariate analysis of Numerical features Let us plot median of the numerical values. Why not mean? because we have already seen there are many outliers and mean is very much influenced by outliers

In [38]:
```
sns.set(font_scale=1.3)
fig, axes = plt.subplots(4, 2, figsize=(18, 20))
axes = [ax for axes_row in axes for ax in axes_row]

for i, c in enumerate(num_cols):
    train.groupby(target_col)[c].median().plot(kind = 'barh', title=f'Median_{
```



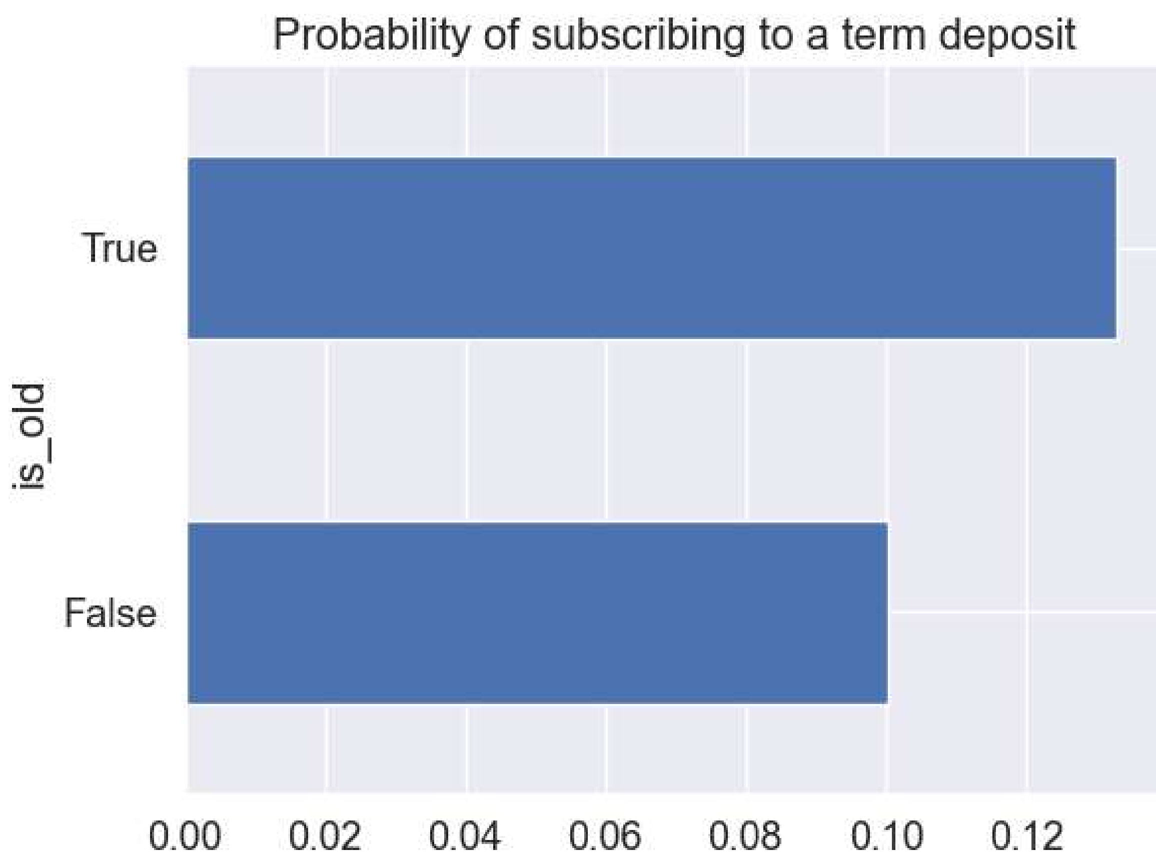

```
Observations

Higher the bank balance more likely to subscribe
Higher the last contacted call duration more likely to subscribe
Let's have closed look at customer age
```

In [40]:
```
#create a new column called is_old and fill with true
train['is_old'] = True

#in each row see of age is less 50
#if yes make old_age value as False fo that row
train.loc[train['customer_age'] <= 50, 'is_old'] = False

#group by old_age and plot the count
_ = train.groupby('is_old')[target_col].mean().sort_values().plot(kind = 'barh
```

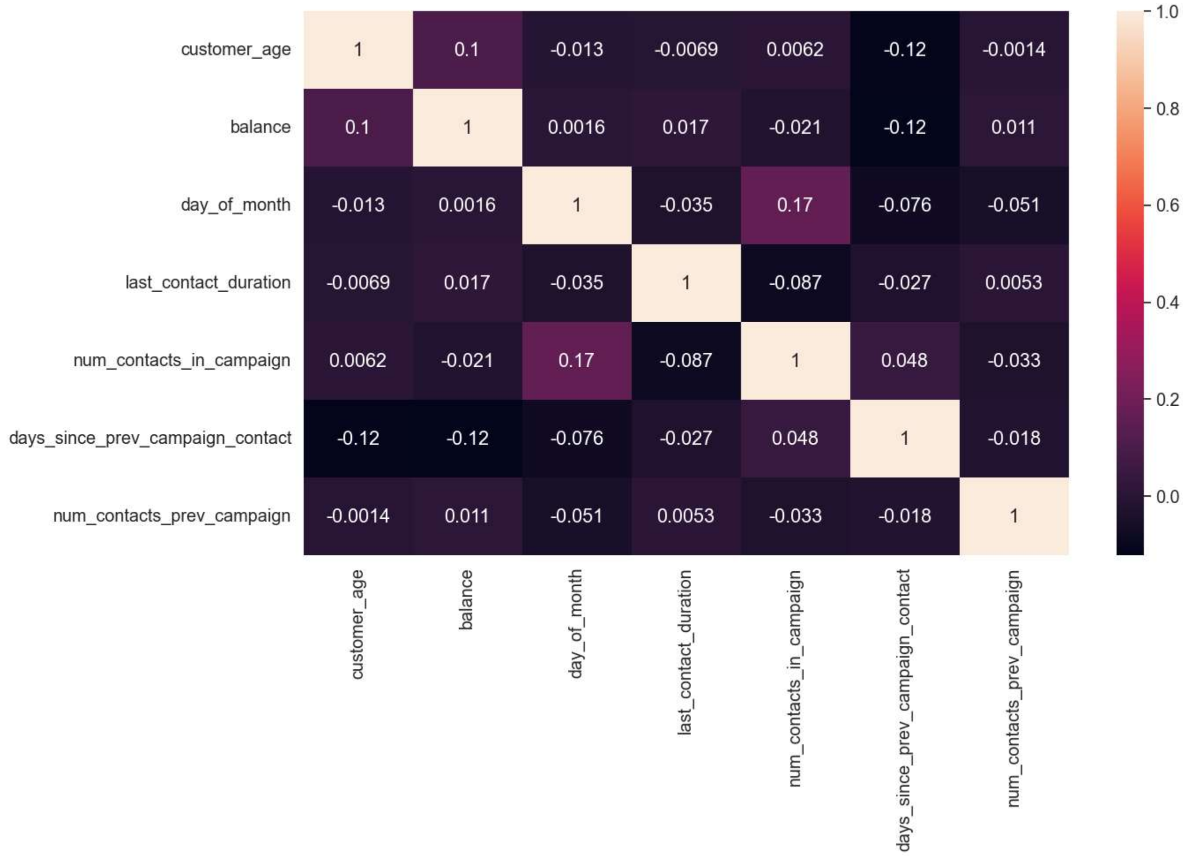



```
Older people are more likely to take the term deposit subscription
```

In [41]:
```
#old_age column is no longer needed
train=train.drop(['is_old'],axis=1)
```

In [42]:
```
#Let's check corelation

plt.figure(figsize=(14, 8))
_ = sns.heatmap(train[num_cols].corr(), annot=True)
```



In [ ]:

Thank you

In [ ]: