MACHINE LEARNING - MINI PROJECT

TOPIC: CUSTOMER BUYING HABITS IN BANKING DOMAIN

MEMBERS:

> UMANG KIRIT LODAYA 60009200032

> JAY BHANUSHALI 60009200047

BATCH/DIV: K - K2 BATCH

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, c
all drive.mount("/content/drive", force_remount=True).
```

LOADING THE DATASETS

TRAINING DATASET

In [2]:

```
path1 = '/content/drive/MyDrive/ML MINI PROJECT/Train.xlsx'
train_data = pd.read_excel(path1)
train_data
```

Out[2]:

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

31647 rows × 18 columns

TESTING DATASET

In [3]:

```
path2 = '/content/drive/MyDrive/ML MINI PROJECT/Test.xlsx'
test_data = pd.read_excel(path2)
test_data
```

Out[3]:

	id	customer_age	job_type	marital	education	default	balance	housing_
0	id_17231	55.0	retired	married	tertiary	no	7136.0	
1	id_34508	24.0	blue-collar	single	secondary	no	179.0	
2	id_44504	46.0	technician	divorced	secondary	no	143.0	
3	id_174	56.0	housemaid	single	unknown	no	6023.0	
4	id_2115	62.0	retired	married	secondary	no	2913.0	
...	...	...	...	...	...	...	...	
13559	id_42406	29.0	management	single	tertiary	no	717.0	
13560	id_14483	NaN	blue-collar	married	secondary	no	604.0	
13561	id_43066	45.0	blue-collar	married	primary	no	237.0	
13562	id_18375	52.0	admin.	married	primary	no	241.0	
13563	id_12898	51.0	technician	married	unknown	no	368.0	

13564 rows × 17 columns

In [4]:

```
train_data.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
```

In [5]:

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13564 entries, 0 to 13563
Data columns (total 17 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   id                                13564 non-null  object 
 1   customer_age                      13294 non-null  float64
 2   job_type                          13564 non-null  object 
 3   marital                           13483 non-null  object 
 4   education                         13564 non-null  object 
 5   default                           13564 non-null  object 
 6   balance                           13383 non-null  float64
 7   housing_loan                      13564 non-null  object 
 8   personal_loan                     13490 non-null  object 
 9   communication_type                13564 non-null  object 
 10  day_of_month                      13564 non-null  int64  
 11  month                             13564 non-null  object 
 12  last_contact_duration             13442 non-null  float64
 13  num_contacts_in_campaign          13519 non-null  float64
 14  days_since_prev_campaign_contact  2441 non-null   float64
 15  num_contacts_prev_campaign        13564 non-null  int64  
 16  prev_campaign_outcome             13564 non-null  object 
dtypes: float64(5), int64(2), object(10)
memory usage: 1.8+ MB
```

PREPROCESSING

HANDLING DUPLICATE VALUES

In [6]:

```
duplicate = train_data[train_data.duplicated(keep='first')]
print("DUPLICATE ROWS IN TRAINING DATASET:",duplicate.shape)
```

```
DUPLICATE ROWS IN TRAINING DATASET: (0, 18)
```

In [7]:

```
duplicate = test_data[test_data.duplicated(keep='first')]
print("DUPLICATE ROWS IN TESTING DATASET:",duplicate.shape)
```

```
DUPLICATE ROWS IN TESTING DATASET: (0, 17)
```

THUS, THERE ARE NO DUPLICATE VALUES IN BOTH DATASETS

HANDLING NULL VALUES IN TRAINING DATA

In [8]:

```
# GETTING THE NULL VALUES
train_data.isnull().sum()
```

Out[8]:

```
id                                      0
customer_age                          619
job_type                                0
marital                               150
education                               0
default                                 0
balance                               399
housing_loan                            0
personal_loan                         149
communication_type                      0
day_of_month                            0
month                                   0
last_contact_duration                 311
num_contacts_in_campaign              112
days_since_prev_campaign_contact    25831
num_contacts_prev_campaign              0
prev_campaign_outcome                   0
term_deposit_subscribed                 0
dtype: int64
```

AS WE CAN SEE, **days_since_prev_campaign_contact** COLUMN HAS 80% NULL VALUE.

THUS, WE ARE DROPPING THIS COLUMN

SINCE OUR DATA SAMPLE ARE INDEPENDENT AND IDENTICALLY DISTRIBUTED (IID), THUS WE REMOVE THE **id** COLUMN, GIVEN THAT ALL SAMPLES COME FROM OR REFER TO THE SAME SOURCE/OBJECT AND THEY DON'T SOMEHOW IDENTIFY THE SAMPLE CLASS.

In [9]:

```
# DROPPING COLUMN
train_data.drop(['days_since_prev_campaign_contact'], axis=1, inplace=True)

# DROPPING THE id COLUMN
train_data.drop(['id'], axis=1, inplace=True)
train_data
```

Out[9]:

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

31647 rows × 16 columns

In [10]:

```
# GETTING NULL VALUES
train_data.isnull().sum()
```

Out[10]:

```
customer_age                 619
job_type                       0
marital                      150
education                      0
default                        0
balance                      399
housing_loan                   0
personal_loan                149
communication_type             0
day_of_month                   0
month                          0
last_contact_duration        311
num_contacts_in_campaign     112
num_contacts_prev_campaign     0
prev_campaign_outcome          0
term_deposit_subscribed        0
dtype: int64
```

FILLING NULL VALUES WITH MODE IN COLUMNS WITH TYPE OBJECT

In [11]:

```
obj = train_data.select_dtypes(include='object')
obj.isna().sum()
```

Out[11]:

```
job_type                   0
marital                  150
education                  0
default                    0
housing_loan               0
personal_loan            149
communication_type         0
month                      0
prev_campaign_outcome      0
dtype: int64
```

marital AND personal_loan ARE dtype: object COLUMNS WITH NULL VALUES WHICH WILL BE FILLED BY MODE VALUE

In [12]:

```
# GETTING MODE VALUE OF MARITAL STATUS
x = train_data['marital'].mode()
x
```

Out[12]:

```
0    married
dtype: object
```

In [13]:

```
# FILLING NULL VALUES WITH MODE IN marital
train_data['marital'].fillna(x[0], inplace=True)
```

In [14]:

```
# GETTING MODE VALUE OF personal_loan
x = train_data['personal_loan'].mode()
x
```

Out[14]:

```
0    no
dtype: object
```

In [15]:

```
# FILLING NULL VALUES WITH MODE IN personal_loan
train_data['personal_loan'].fillna(x[0], inplace=True)
```

FILLING NULL VALUES IN NUMERICAL COLUMNS

In [16]:

```
float_cols = train_data.select_dtypes(include='float64')
float_cols.isna().sum()
```

Out[16]:

```
customer_age                619
balance                     399
last_contact_duration       311
num_contacts_in_campaign    112
dtype: int64
```

CHECKING FOR OUTLIERS IN NUMERICAL COLUMNS USING BOXPLOT

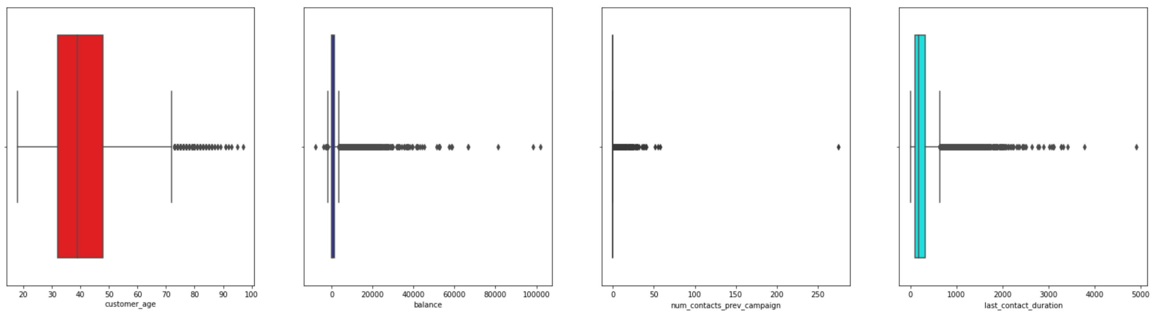
In [17]:

```
fig, axes = plt.subplots(1, 4, figsize=(28, 7), sharey=True)
sns.boxplot(train_data['customer_age'], ax=axes[0], color='r')
sns.boxplot(train_data['balance'], ax=axes[1], color='b')
sns.boxplot(train_data['num_contacts_prev_campaign'], ax=axes[2], color='y')
sns.boxplot(train_data['last_contact_duration'], ax=axes[3], color='cyan')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f80db581810>
```



THE OUTLIERS CAN HIGHLY DEVIATE THE MEAN OF THE COLUMNS HENCE IT IS MORE APPROPRIATE TO REPLACE THE NULL VALUES WITH MEDIAN

In [18]:

```
# GETTING MEDIAN AGE AS AGE IS FLOATING VALUE
x = train_data['customer_age'].median()
x
```

Out[18]:

```
39.0
```

In [19]:

```
# FILLING NULL VALUES IN customer_age COLUMN WITH MEDIAN
train_data['customer_age'].fillna(x, inplace=True)
```

In [20]:

```
# GETTING MEDIAN BALANCE VALUE
x = train_data['balance'].median()
x
```

Out[20]:

```
449.0
```

In [21]:

```
# FILLING NULL VALUES IN balance COLUMN WITH MEDIAN
train_data['balance'].fillna(x, inplace=True)
```

In [22]:

```
# GETTING MEDIAN VALUE OF last_call_duration COLUMN
x = train_data['last_contact_duration'].median()
x
```

Out[22]:

```
179.0
```

In [23]:

```
# FILLING NULL VALUES IN last_contact_duration COLUMN WITH MEDIAN
train_data['last_contact_duration'].fillna(x, inplace=True)
```

In [24]:

```
# GETTING MEDIAN VALUE OF num_contact_in_campaign COLUMN
x = train_data['num_contacts_in_campaign'].median()
x
```

Out[24]:

```
2.0
```

In [25]:

```
# FILLING NULL VALUES IN num_contacts_in_campaign COLUMN WITH MEDIAN AS MEAN VALUE CAN
 BE INFLUENCED BY OUTLIER, BUT MEDIAN WILL NOT
train_data['num_contacts_in_campaign'].fillna(x, inplace=True)
```

In [26]:

```
train_data.isnull().sum()
```

Out[26]:

```
customer_age                 0
job_type                     0
marital                      0
education                    0
default                      0
balance                      0
housing_loan                 0
personal_loan                0
communication_type           0
day_of_month                 0
month                        0
last_contact_duration        0
num_contacts_in_campaign     0
num_contacts_prev_campaign   0
prev_campaign_outcome        0
term_deposit_subscribed      0
dtype: int64
```

THUS, WE HAVE HANDLED ALL NULL VALUES IN TRAINING DATA SET

HANDLING NULL VALUES IN TESTING DATA

AS WE HAVE DROPPED THE FOLLOWING COLUMNS FROM TRAIN DATA WE HAVE TO DROP THEM FROM TEST DATA AS WELL:

- id
- days_since_prev_campaign_contact

In [27]:

```
# DROPPING COLUMN
test_data.drop(['days_since_prev_campaign_contact'], axis=1, inplace=True)

# DROPPING THE id COLUMN
test_data.drop(['id'], axis=1, inplace=True)
test_data
```

Out[27]:

	customer_age	job_type	marital	education	default	balance	housing_loan	pers
0	55.0	retired	married	tertiary	no	7136.0	no	
1	24.0	blue-collar	single	secondary	no	179.0	yes	
2	46.0	technician	divorced	secondary	no	143.0	no	
3	56.0	housemaid	single	unknown	no	6023.0	no	
4	62.0	retired	married	secondary	no	2913.0	no	
...	...	...	...	...	...	...	...	
13559	29.0	management	single	tertiary	no	717.0	yes	
13560	NaN	blue-collar	married	secondary	no	604.0	yes	
13561	45.0	blue-collar	married	primary	no	237.0	yes	
13562	52.0	admin.	married	primary	no	241.0	yes	
13563	51.0	technician	married	unknown	no	368.0	yes	

13564 rows × 15 columns

In [28]:

```
# GETTING COUNT OF ALL NULL VALUES
test_data.isnull().sum()
```

Out[28]:

```
customer_age                270
job_type                      0
marital                      81
education                     0
default                       0
balance                     181
housing_loan                  0
personal_loan                74
communication_type            0
day_of_month                  0
month                         0
last_contact_duration       122
num_contacts_in_campaign     45
num_contacts_prev_campaign    0
prev_campaign_outcome         0
dtype: int64
```

In [29]:

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13564 entries, 0 to 13563
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   customer_age                 13294 non-null  float64
 1   job_type                     13564 non-null  object 
 2   marital                      13483 non-null  object 
 3   education                    13564 non-null  object 
 4   default                      13564 non-null  object 
 5   balance                      13383 non-null  float64
 6   housing_loan                 13564 non-null  object 
 7   personal_loan                13490 non-null  object 
 8   communication_type           13564 non-null  object 
 9   day_of_month                 13564 non-null  int64  
 10  month                        13564 non-null  object 
 11  last_contact_duration        13442 non-null  float64
 12  num_contacts_in_campaign     13519 non-null  float64
 13  num_contacts_prev_campaign   13564 non-null  int64  
 14  prev_campaign_outcome        13564 non-null  object 
dtypes: float64(4), int64(2), object(9)
memory usage: 1.6+ MB
```

In [30]:

```
test_data.select_dtypes(include="int64").isnull().sum()
```

Out[30]:

```
day_of_month                 0
num_contacts_prev_campaign   0
dtype: int64
```

WE DO NOT HAVE ANY NULL VALUES IN INT64 TYPE COLUMNS

FILLING NULL VALUE WITH MODE IN COLUMNS WITH DTYPE OBJECT

In [31]:

```
# displaying object columns with null values
test_data.select_dtypes(include="object").isnull().sum()
```

Out[31]:

```
job_type                0
marital                81
education               0
default                 0
housing_loan            0
personal_loan          74
communication_type      0
month                   0
prev_campaign_outcome   0
dtype: int64
```

WE HAVE TO REPLACE MARITAL AND PERSONAL_LOAN WITH THE MODE OF RESPECTIVE COLUMNS

In [32]:

```
for col in ['marital', 'personal_loan']:
  col_mode = test_data[col].mode()[0]
  test_data[col].fillna(col_mode, inplace=True)

test_data.select_dtypes(include="object").isnull().sum()
```

Out[32]:

```
job_type                0
marital                 0
education               0
default                 0
housing_loan            0
personal_loan           0
communication_type      0
month                   0
prev_campaign_outcome   0
dtype: int64
```

FILLING NULL VALUE WITH MEDIAN IN COLUMNS DTYPE FLOAT64

In [33]:

```
# displaying columns with type float64 having null values
test_data.select_dtypes(include="float64").isnull().sum()
```

Out[33]:

```
customer_age               270
balance                    181
last_contact_duration      122
num_contacts_in_campaign    45
dtype: int64
```

In [34]:

```
for col in ['customer_age', 'balance', 'last_contact_duration', 'num_contacts_in_campai
gn']:
  col_median = test_data[col].median()
  test_data[col].fillna(col_median, inplace=True)

test_data.select_dtypes(include="float64").isnull().sum()
```

Out[34]:

```
customer_age               0
balance                    0
last_contact_duration      0
num_contacts_in_campaign   0
dtype: int64
```

NOW WE ARE RID OF ALL NULL VALUES FROM TRAINING AS WELL AS TESTING DATASET

In [35]:

```
test_data.isnull().sum()
```

Out[35]:

```
customer_age                0
job_type                    0
marital                     0
education                   0
default                     0
balance                     0
housing_loan                0
personal_loan               0
communication_type          0
day_of_month                0
month                       0
last_contact_duration       0
num_contacts_in_campaign    0
num_contacts_prev_campaign  0
prev_campaign_outcome       0
dtype: int64
```

In [36]:

```
train_data.isnull().sum()
```

Out[36]:

```
customer_age                0
job_type                    0
marital                     0
education                   0
default                     0
balance                     0
housing_loan                0
personal_loan               0
communication_type          0
day_of_month                0
month                       0
last_contact_duration       0
num_contacts_in_campaign    0
num_contacts_prev_campaign  0
prev_campaign_outcome       0
term_deposit_subscribed     0
dtype: int64
```

GETTING INFORMATION ON TRAINING DATA

In [37]:

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31647 entries, 0 to 31646
Data columns (total 16 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   customer_age                31647 non-null  float64
 1   job_type                    31647 non-null  object
 2   marital                     31647 non-null  object
 3   education                   31647 non-null  object
 4   default                     31647 non-null  object
 5   balance                     31647 non-null  float64
 6   housing_loan                31647 non-null  object
 7   personal_loan               31647 non-null  object
 8   communication_type          31647 non-null  object
 9   day_of_month                31647 non-null  int64
 10  month                       31647 non-null  object
 11  last_contact_duration       31647 non-null  float64
 12  num_contacts_in_campaign    31647 non-null  float64
 13  num_contacts_prev_campaign  31647 non-null  int64
 14  prev_campaign_outcome       31647 non-null  object
 15  term_deposit_subscribed     31647 non-null  int64
dtypes: float64(4), int64(3), object(9)
memory usage: 3.9+ MB
```

COLUMNS WITH object AS Dtype (CATEGORICAL COLUMNS) ARE:

1. job_type
2. marital
3. education
4. default
5. housing_loan
6. personal_loan
7. communication_type
8. month
9. prev_campaign_outcome

PRINTING NUMBER OF UNIQUE VALUES OF THE CATEGORICAL COLUMNS

In [38]:

```
# GETTING UNIQUE VALUES OF 'marital' COLUMN
train_data['marital'].value_counts()
```

Out[38]:

```
married     19095
single       8857
divorced     3695
Name: marital, dtype: int64
```

In [39]:

```
# GETTING UNIQUE VALUES OF 'job_type' COLUMN
train_data['job_type'].value_counts()
```

Out[39]:

```
blue-collar      6816
management       6666
technician       5220
admin.           3627
services         2923
retired          1591
self-employed    1111
entrepreneur     1037
unemployed        901
housemaid         893
student           663
unknown           199
Name: job_type, dtype: int64
```

In [40]:

```
# GETTING UNIQUE VALUES OF 'education' COLUMN
train_data['education'].value_counts()
```

Out[40]:

```
secondary    16247
tertiary      9321
primary       4787
unknown       1292
Name: education, dtype: int64
```

In [41]:

```
# GETTING UNIQUE VALUES OF 'default' COLUMN
train_data['default'].value_counts()
```

Out[41]:

```
no     31094
yes      553
Name: default, dtype: int64
```

In [42]:

```
# GETTING UNIQUE VALUES OF 'housing_loan' COLUMN
train_data['housing_loan'].value_counts()
```

Out[42]:

```
yes    17700
no     13947
Name: housing_loan, dtype: int64
```

In [43]:

```
# GETTING UNIQUE VALUES OF 'personal_loan' COLUMN
train_data['personal_loan'].value_counts()
```

Out[43]:

```
no     26612
yes     5035
Name: personal_loan, dtype: int64
```

In [44]:

```
# GETTING UNIQUE VALUES OF 'communication_type' COLUMN
train_data['communication_type'].value_counts()
```

Out[44]:

```
cellular     20480
unknown       9151
telephone     2016
Name: communication_type, dtype: int64
```

In [45]:

```
# GETTING UNIQUE VALUES OF 'month' COLUMN
train_data['month'].value_counts()
```

Out[45]:

```
may    9685
jul    4786
aug    4308
jun    3746
nov    2801
apr    2111
feb    1836
jan     953
oct     510
sep     417
mar     338
dec     156
Name: month, dtype: int64
```

In [46]:

```
# GETTING UNIQUE VALUES OF 'prev_campaign_outcome' COLUMN
train_data['prev_campaign_outcome'].value_counts()
```

Out[46]:

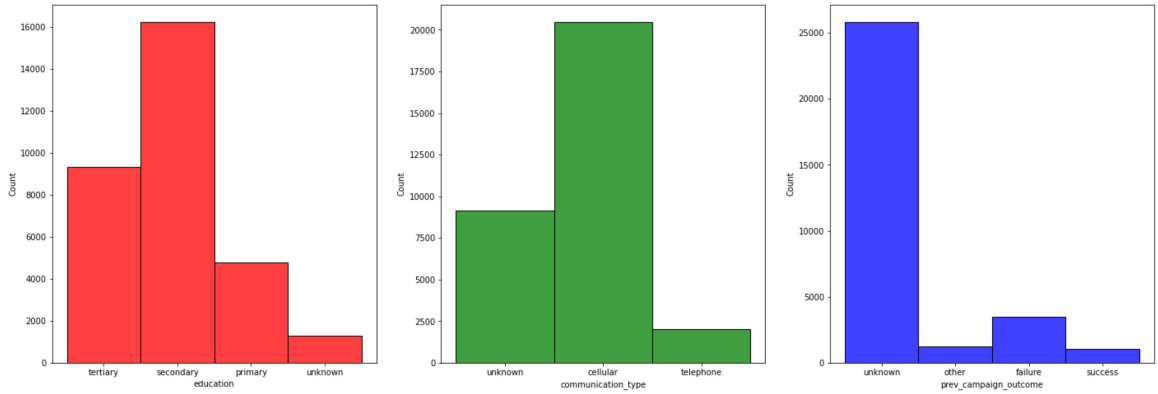
```
unknown    25833
failure     3472
other       1272
success     1070
Name: prev_campaign_outcome, dtype: int64
```

In [47]:

```
# PLOT HISTOGRAM OF FEATURES
fig, axes = plt.subplots(1, 3, figsize=(24, 8), sharey=False)
sns.histplot(train_data, ax=axes[0], x="education", color='r')
sns.histplot(train_data, ax=axes[1], x="communication_type", color='g')
sns.histplot(train_data, ax=axes[2], x="prev_campaign_outcome", color='b')
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f80db4fd6d0>
```

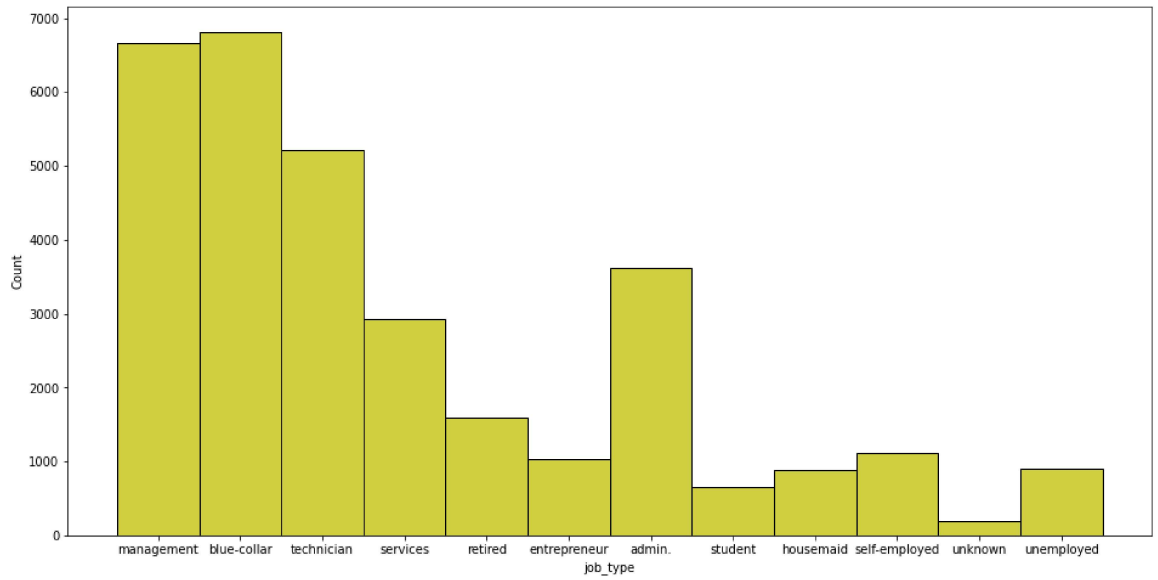


In [48]:

```
fig, axes = plt.subplots(1, 1, figsize=(16, 8))
sns.histplot(train_data, x="job_type", color='y')
```

Out[48]:

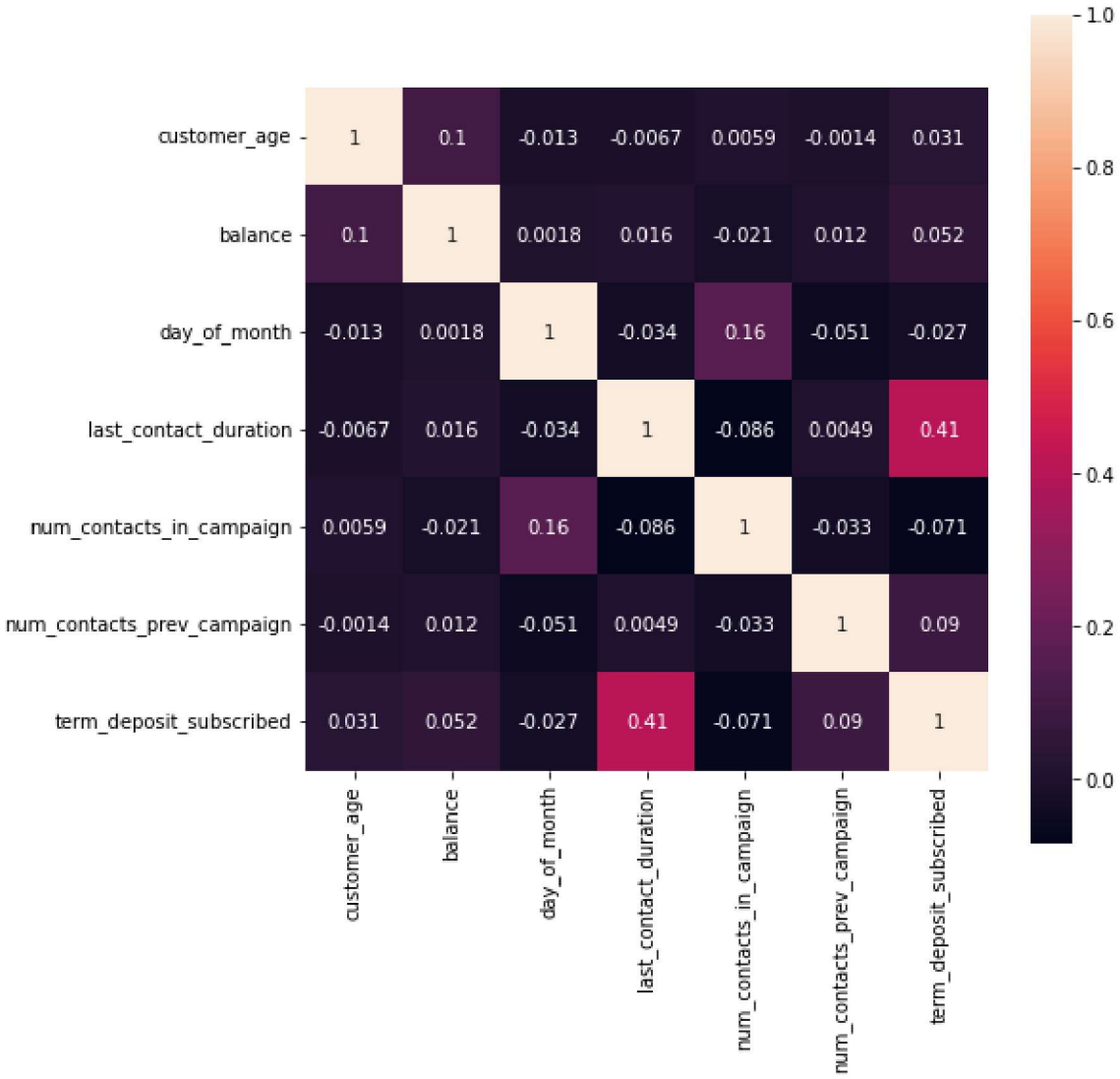
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f80db59b450>
```



CORRELATION HEATMAP

In [49]:

```
corr = train_data.corr()
fig = plt.figure(figsize = (8, 8))
sns.heatmap(corr, annot=True, square=True)
plt.show()
```



- **WE CAN SEE A SLIGHT CORRELATION BETWEEN `LAST_CONTACT_DURATION` AND `TERM_DEPOSIT_SCBSCRIBED` (LABEL)**
- **APART FROM THIS THERE IS NO SIGNIFICANT CORRELATION AMONG THE FEATURES**

LABEL ENCODING

In [50]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

ENCODING IN TRAINING DATASET

In [51]:

```
# LABEL ENCODING THE CATEGORICAL FEATURES AND LABEL
obj_cols = train_data.select_dtypes(include='object').columns
print("OBJECT COLUMNS TO BE ENCODED ARE:\n", list(obj_cols))
for col in obj_cols:
    train_data[col] = le.fit_transform(train_data[col])

train_data.head(10)
```

```
OBJECT COLUMNS TO BE ENCODED ARE:
 ['job_type', 'marital', 'education', 'default', 'housing_loan', 'personal
_loan', 'communication_type', 'month', 'prev_campaign_outcome']
```

Out[51]:

	customer_age	job_type	marital	education	default	balance	housing_loan	personal_loan
0	28.0	4	2	2	0	285.0	1	0
1	34.0	1	1	1	0	934.0	0	1
2	46.0	9	1	1	0	656.0	0	0
3	34.0	7	2	1	0	2.0	1	0
4	41.0	1	1	0	0	1352.0	1	0
5	65.0	5	1	0	0	2880.0	0	0
6	57.0	1	1	0	0	495.0	0	0
7	37.0	4	1	2	0	650.0	1	0
8	29.0	9	1	1	0	265.0	1	0
9	50.0	1	1	1	0	407.0	1	1

ENCODING IN TESTING DATASET

In [52]:

```
# LABEL ENCODING THE CATEGORICAL FEATURES AND LABEL
obj_cols = test_data.select_dtypes(include='object').columns
print("OBJECT COLUMNS TO BE ENCODED ARE:\n", list(obj_cols))
for col in obj_cols:
    test_data[col] = le.fit_transform(test_data[col])

test_data.head(10)
```

```
OBJECT COLUMNS TO BE ENCODED ARE:
 ['job_type', 'marital', 'education', 'default', 'housing_loan', 'personal
_loan', 'communication_type', 'month', 'prev_campaign_outcome']
```

Out[52]:

	customer_age	job_type	marital	education	default	balance	housing_loan	personal_loan
0	55.0	5	1	2	0	7136.0	0	0
1	24.0	1	2	1	0	179.0	1	0
2	46.0	9	0	1	0	143.0	0	0
3	56.0	3	2	3	0	6023.0	0	0
4	62.0	5	1	1	0	2913.0	0	0
5	35.0	6	2	2	0	355.0	0	0
6	27.0	3	2	1	0	718.0	1	0
7	29.0	4	2	2	0	54.0	1	1
8	50.0	3	1	1	0	3815.0	0	0
9	37.0	4	2	2	0	52.0	1	1

LABEL ENCODED FOR BOTH THE DATASETS

PREPROCESSING FOR BOTH TRAINING & TESTING DATASETS IS COMPLETED