mrshk7qvz

May 31, 2023

[1]:
```
# Importing Data Analysis Librarys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from scipy.stats import mode
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
```

[2]:
```
bank = pd.read_csv('bank-additional-full.csv', sep = ';')
#Converting dependent variable categorical to dummy
y = pd.get_dummies(bank['y'], columns = ['y'], prefix = ['y'], drop_first =␣
 ↪True)
bank.head()
```

[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	…	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	…	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	…	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	…	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	…	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	…	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

```
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

[3]:
```
bank.info()
```

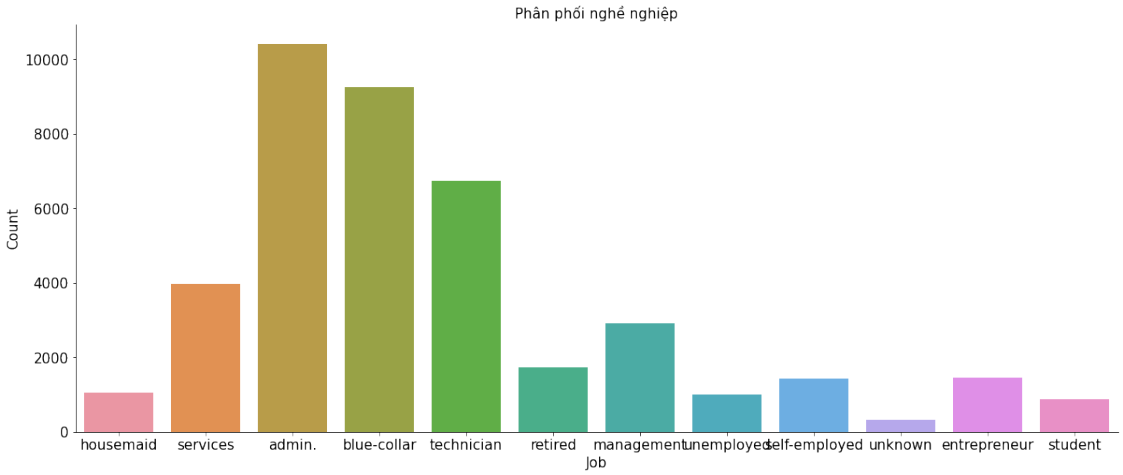
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

[4]:
```
#Đầu tiên, bắt đầu làm việc với các giá trị còn thiếu trong bộ data.
#Trong tập dữ liệu không có giá trị bị thiếu nhưng có một số giá trị "Unknown"
#thực sự là giá trị bị thiếu đối với chúng tôi. Sau khi nghiên cứu bộ dữ liệu␣
 ↪nhóm quyết định để các giá trị unknown
#trong các biến và xử missing value và tiến hành xử lý chúng bằng mode

# TẠO DANH SÁCH CÁC TÍNH NĂNG ĐƯỢC PHÂN LOẠI HOẶC SỐ
Category_list = []
Numerical_list = []

for i in bank.columns:
  if bank[i].dtype == 'object':
```

```
    Category_list.append(i)
  else:
    Numerical_list.append(i)

print('Categorical list:', Category_list)
print('Numerical list :', Numerical_list)
```

```
Categorical list: ['job', 'marital', 'education', 'default', 'housing', 'loan',
'contact', 'month', 'day_of_week', 'poutcome', 'y']
Numerical list : ['age', 'duration', 'campaign', 'pdays', 'previous',
'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']
```

[5]:
```
# Vẽ biểu đồ để có cái nhìn cụ thể về các công việc của những khách hàng của␣
↪ngân hàng
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'job', data = bank)
ax.set_xlabel('Job', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Phân phối nghề nghiệp', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```



[6]:
```
#Ta thấy Admin là công việc chiếm nhiều nhất nên xử lý unknown thay thế bằng␣
↪admin.
#Xử lý missing value
mode_job = bank['job'].mode()
bank['job'] = bank['job'].replace({'admin':'admin.'})
```

[7]:
```
#Tương tự kiểm tra xem trong tất cả các cột của bank cột nào có unknown thì␣
 ↪thay thế bằng mode của cột đó
for column in bank.columns:
    if 'unknown' in bank[column].values:
        mode_value = mode(bank[column])[0][0]  # Lấy giá trị mode
        bank[column].replace('unknown', mode_value, inplace=True)
```

[8]:
```
#Quan sát một số biến khác
bank.default.value_counts()
```

[8]:
```
no     41185
yes        3
Name: default, dtype: int64
```

[9]:
```
#Vì default feature hầu hết là "no", điều này rất mất cân bằng, nên bỏ tính␣
 ↪năng này.
bank.drop("default",inplace=True,axis=1)
```

[10]:
```
#Đối với tính năng education, có một số giá trị tương tự như basic.9y, basic.6y␣
 ↪và basic.4y.
#được chuyển đổi chúng thành "mid.shool
lst=['basic.9y','basic.6y','basic.4y']
for i in lst:
    bank.loc[bank['education'] == i, 'education'] = "mid.school"

bank['education'].value_counts()
```

[10]:
```
university.degree      13899
mid.school             12513
high.school             9515
professional.course     5243
illiterate                18
Name: education, dtype: int64
```

[11]:
```
#Ngoài ra còn có một vấn đề trong tính năng pdays.
#nếu giá trị là 999, thì nó sẽ được thay thế bằng 0, nghĩa là khách hàng chưa␣
 ↪được liên hệ trước đó.
bank.pdays.value_counts()
```

[11]:
```
999    39673
3        439
6        412
4        118
9         64
2         61
7         60
12        58
```

```
10        52
5         46
13        36
11        28
1         26
15        24
14        20
8         18
0         15
16        11
17         8
18         7
22         3
19         3
21         2
25         1
26         1
27         1
20         1
Name: pdays, dtype: int64
```

[12]:
```
bank.loc[bank['pdays'] == 999, 'pdays'] = 0
bank.pdays.value_counts()
```

[12]:
```
0     39688
3       439
6       412
4       118
9        64
2        61
7        60
12       58
10       52
5        46
13       36
11       28
1        26
15       24
14       20
8        18
16       11
17        8
18        7
22        3
19        3
21        2
25        1
```

```
26        1
27        1
20        1
Name: pdays, dtype: int64
```

[13]:
```
# Check whether if missing values exist
bank.isna().sum()
```

[13]:
```
age               0
job               0
marital           0
education         0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

[14]:
```
#Handling Duplicated Data
bank.duplicated().sum()
```

[14]:
```
17
```

[15]:
```
bank.drop_duplicates(inplace=True)
```

[16]:
```
#Quan sát và xử lý ngoại lai cho các biến
age_mean = bank['age'].mean()
age_std = bank['age'].std()
```

[17]:
```
z_scores = (bank['age'] - age_mean) / age_std
```

[18]:
```
threshold = 3 # Chọn ngưỡng cố định
outliers = bank.loc[abs(z_scores) > threshold, 'age']
bank = bank.drop(bank[z_scores > threshold].index)
bank.describe()
```

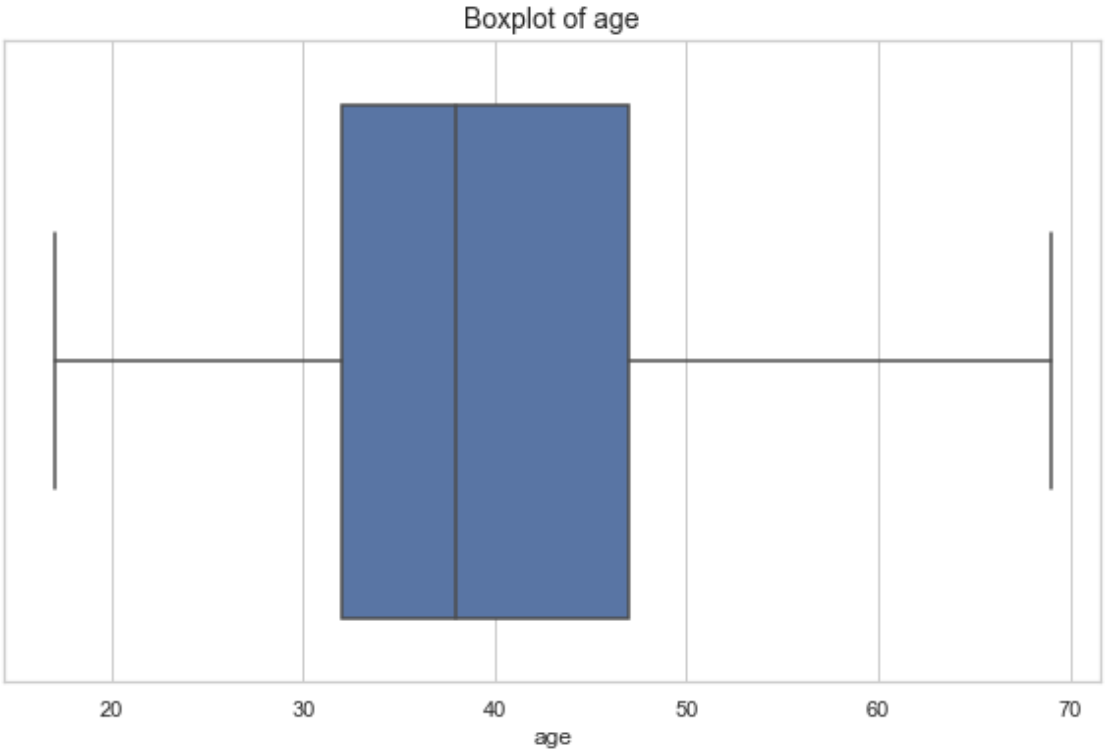
[18]:

	age	duration	campaign	pdays	previous	\
count	40802.000000	40802.000000	40802.000000	40802.000000	40802.000000	
mean	39.673815	258.191584	2.573673	0.212514	0.168129	
std	9.782640	259.738381	2.777806	1.327910	0.485182	
min	17.000000	0.000000	1.000000	0.000000	0.000000	
25%	32.000000	102.000000	1.000000	0.000000	0.000000	
50%	38.000000	179.000000	2.000000	0.000000	0.000000	
75%	47.000000	319.000000	3.000000	0.000000	0.000000	
max	71.000000	4918.000000	56.000000	27.000000	7.000000	

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	40802.000000	40802.000000	40802.000000	40802.000000	40802.000000
mean	0.103875	93.579620	-40.538601	3.646051	5168.283155
std	1.559121	0.575396	4.584466	1.722466	71.262176
min	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	1.400000	94.767000	-26.900000	5.045000	5228.100000

[19]:
```
bank=bank[(bank['age'] >= 17) & (bank['age'] <= 69)]
```

[20]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=bank['age'])
plt.xlabel('age', fontsize=12)
plt.title('Boxplot of age', fontsize=14)
plt.show()
```



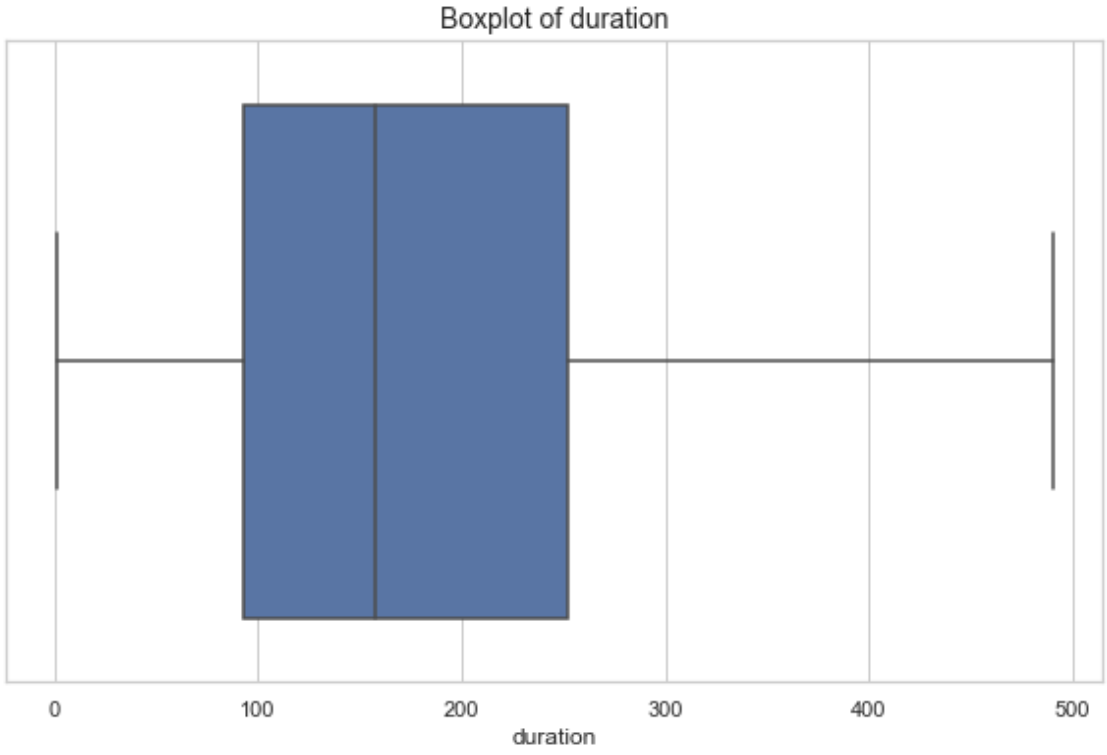
[21]:
```
# Tìm IQR của biến duration
Q1 = bank['duration'].quantile(0.25)
Q3 = bank['duration'].quantile(0.75)
IQR = Q3 - Q1

# Tính giá trị lower và upper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Lấy các giá trị outliers
outliers = bank[(bank['duration'] < lower_bound) | (bank['duration'] >␣
 ↪upper_bound)]

# Loại bỏ các giá trị outliers khỏi bộ dữ liệu
bank = bank[(bank['duration'] >= lower_bound) & (bank['duration'] <=␣
 ↪upper_bound)]
```

[22]:
```
import numpy as np

# Tính toán modified Z-score
median = np.median(bank['duration'])
mad = np.median(np.abs(bank['duration'] - median))
modified_z_scores = 0.6745 * (bank['duration'] - median) / mad
```

```
# Lấy tập dữ liệu ngoại trừ outlier
threshold = 3.5
outliers = bank[abs(modified_z_scores) > threshold]['age']
bank = bank[abs(modified_z_scores) <= threshold]
```

[23]:
```
bank=bank[(bank['duration'] >0) & (bank['duration'] <= 490)]
```

[24]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=bank['duration'])
plt.xlabel('duration', fontsize=12)
plt.title('Boxplot of duration', fontsize=14)
plt.show()
```



[25]:
```
campaign_mean = bank['campaign'].mean()
campaign_std = bank['campaign'].std()
```

[26]:
```
z_scores = (bank['campaign'] - campaign_mean) / campaign_std
```

[27]:
```
threshold = 3 # Chọn ngưỡng cố định
outliers = bank.loc[abs(z_scores) > threshold, 'age']
```

```
bank = bank.drop(bank[z_scores > threshold].index)
```

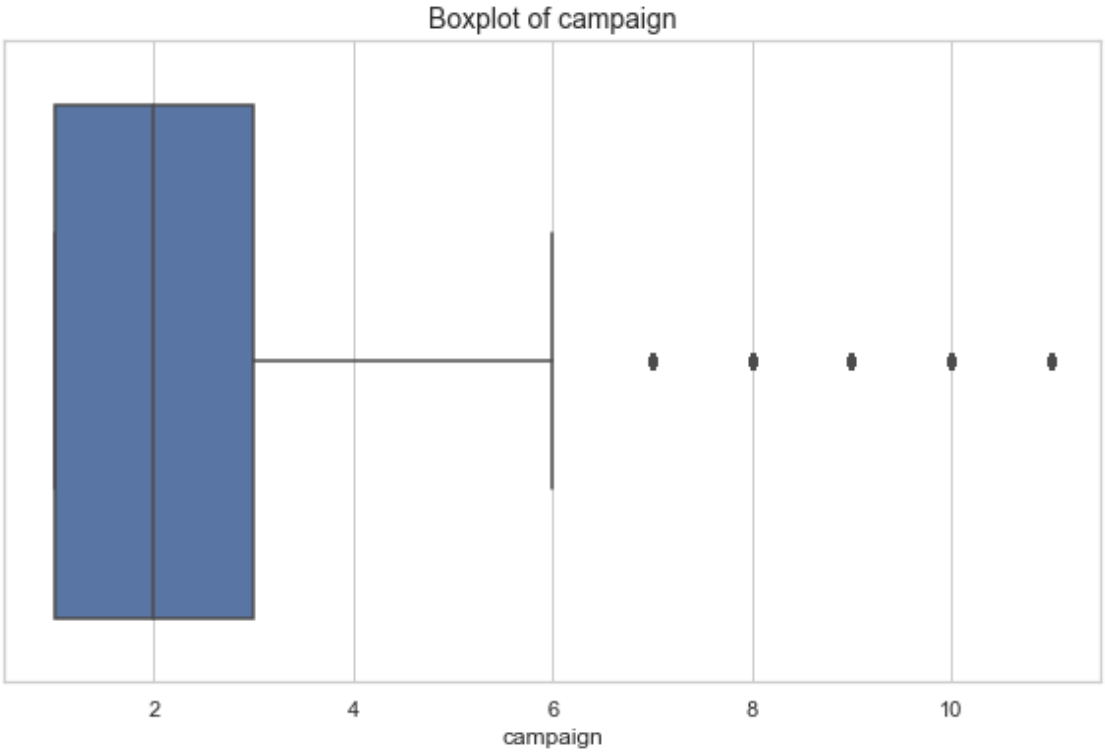
[28]:
```
bank.describe()
```

[28]:

	age	duration	campaign	pdays	previous
count	35026.000000	35026.000000	35026.000000	35026.000000	35026.000000
mean	39.574116	183.547194	2.321475	0.198938	0.169246
std	9.647018	113.452652	1.823020	1.273457	0.483938
min	17.000000	1.000000	1.000000	0.000000	0.000000
25%	32.000000	95.000000	1.000000	0.000000	0.000000
50%	38.000000	159.000000	2.000000	0.000000	0.000000
75%	47.000000	253.000000	3.000000	0.000000	0.000000
max	69.000000	490.000000	11.000000	27.000000	7.000000

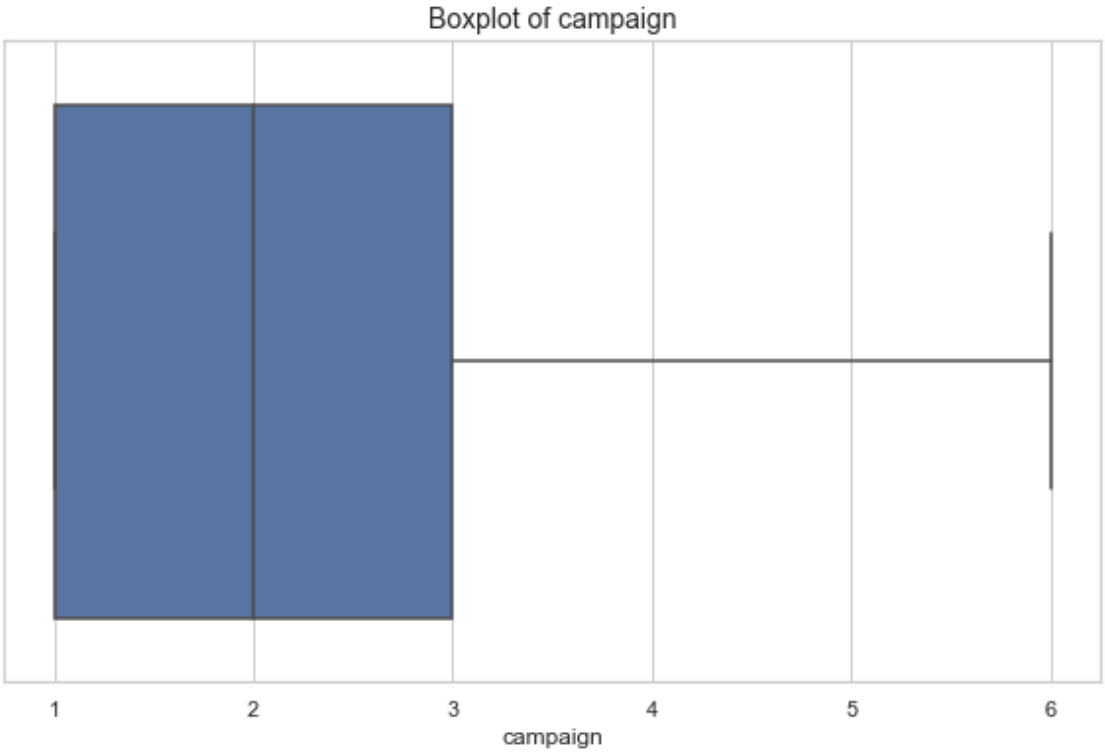
	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	35026.000000	35026.000000	35026.000000	35026.000000	35026.000000
mean	0.095186	93.572596	-40.508311	3.640401	5168.135282
std	1.558845	0.575183	4.589532	1.721586	70.875676
min	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	1.100000	93.444000	-41.800000	4.857000	5191.000000
75%	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	1.400000	94.767000	-26.900000	5.045000	5228.100000

[29]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=bank['campaign'])
plt.xlabel('campaign', fontsize=12)
plt.title('Boxplot of campaign', fontsize=14)
plt.show()
```



[30]:
```
bank=bank[(bank['campaign'] >=1) & (bank['campaign'] <= 6)]
```

[31]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=bank['campaign'])
plt.xlabel('campaign', fontsize=12)
plt.title('Boxplot of campaign', fontsize=14)
plt.show()
```

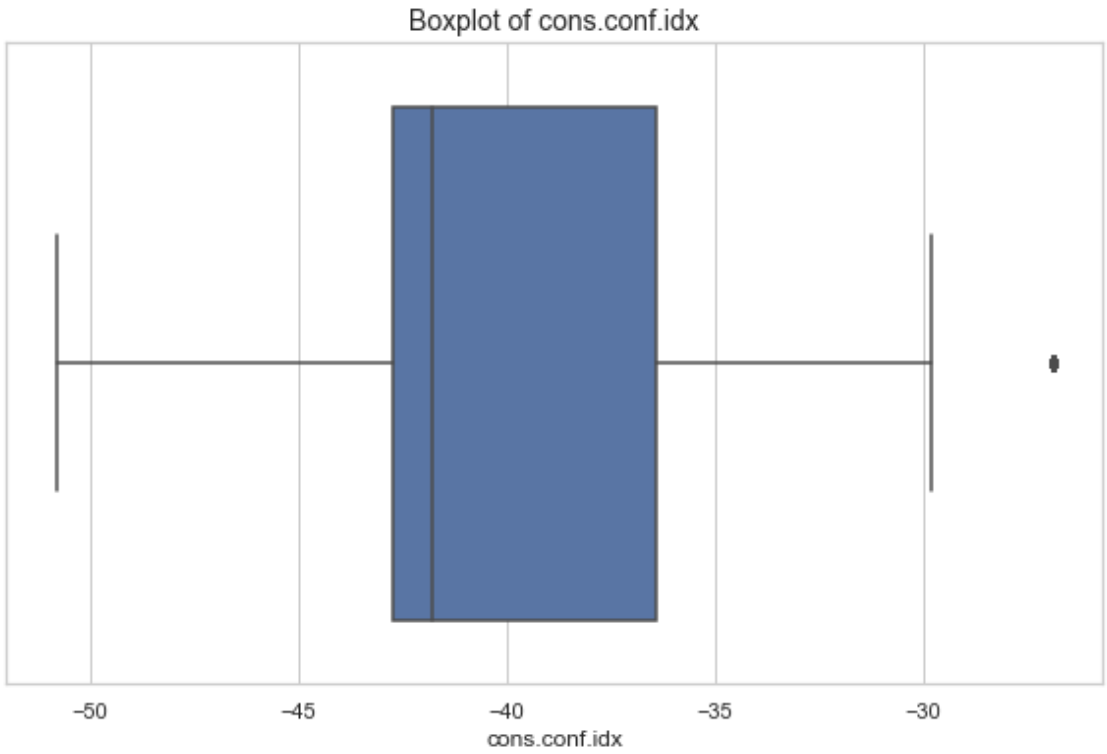


[32]:
```
age_mean = bank['cons.conf.idx'].mean()
age_std = bank['cons.conf.idx'].std()
```

[33]:
```
z_scores = (bank['cons.conf.idx'] - age_mean) / age_std
```

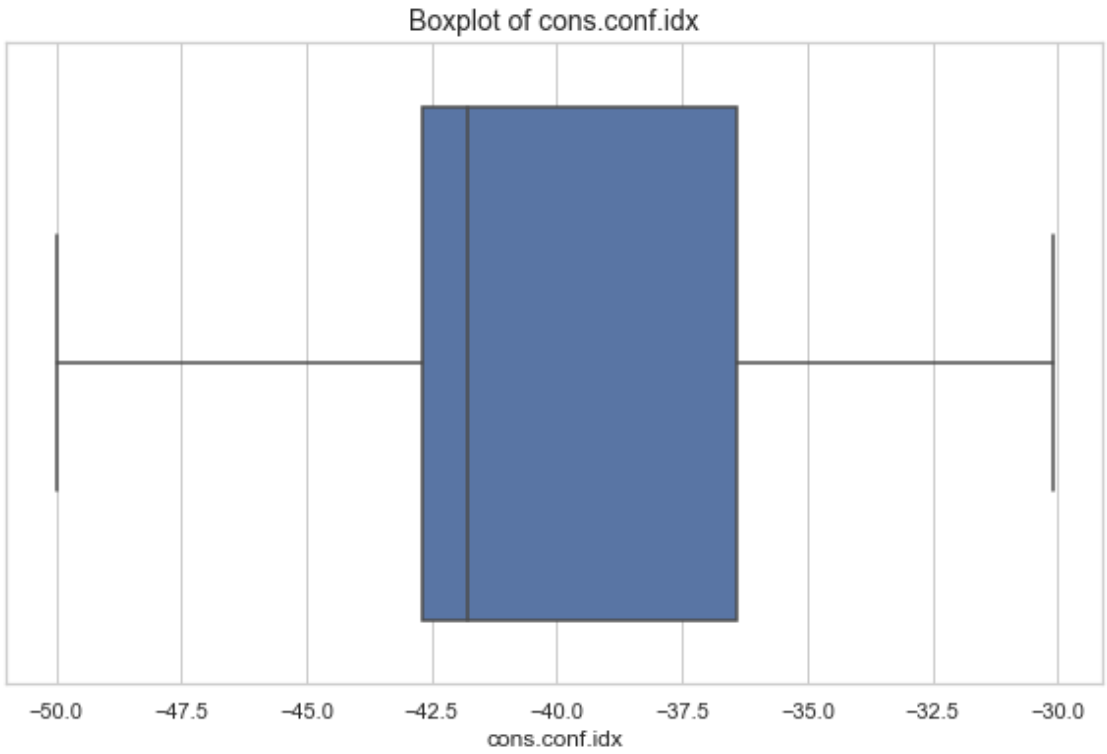
[34]:
```
threshold = 3 # Chọn ngưỡng cố định
outliers = bank.loc[abs(z_scores) > threshold, 'cons.conf.idx']
bank = bank.drop(bank[z_scores > threshold].index)
```

[35]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=bank['cons.conf.idx'])
plt.xlabel('cons.conf.idx', fontsize=12)
plt.title('Boxplot of cons.conf.idx', fontsize=14)
plt.show()
```

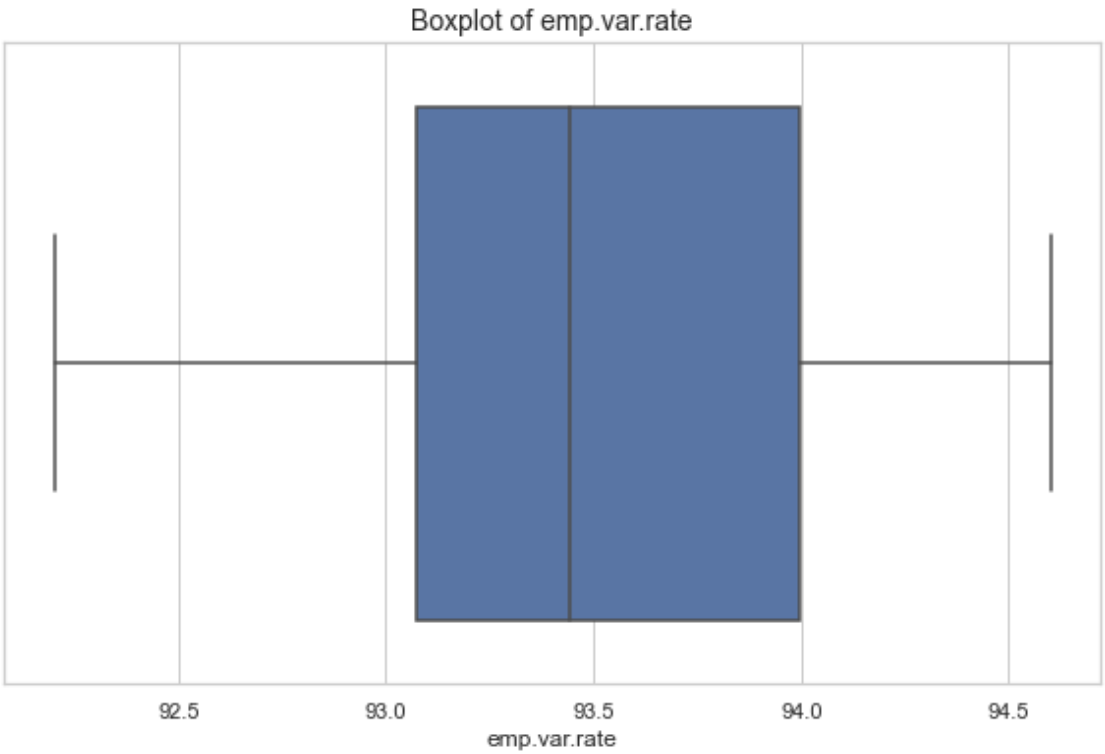


[36]:
```
bank=bank[(bank['cons.conf.idx'] >=-50) & (bank['cons.conf.idx'] <=-30)]
```

[37]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=bank['cons.conf.idx'])
plt.xlabel('cons.conf.idx', fontsize=12)
plt.title('Boxplot of cons.conf.idx', fontsize=14)
plt.show()
```

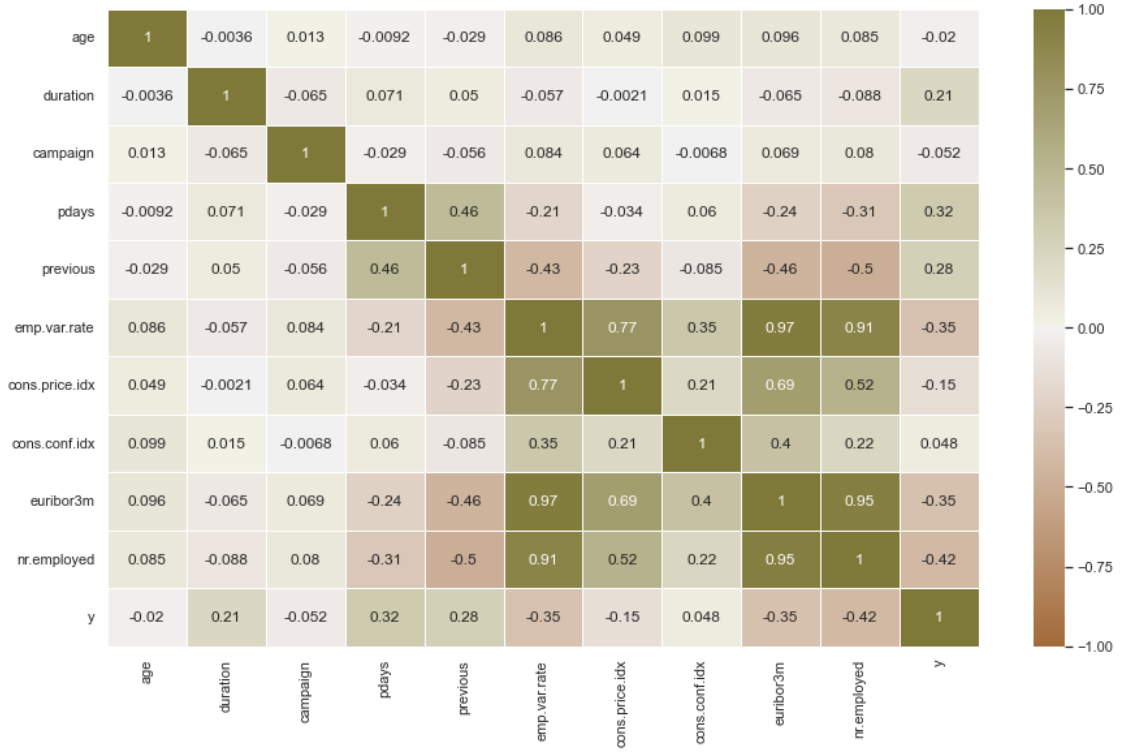


[38]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=bank['cons.price.idx'])
plt.xlabel('emp.var.rate', fontsize=12)
plt.title('Boxplot of emp.var.rate', fontsize=14)
plt.show()
```



[39]:
```
#Chuyển đổi biến mục tiêu thành số
bank.y = bank.y.map({'no':0, 'yes':1}).astype('uint8')
```

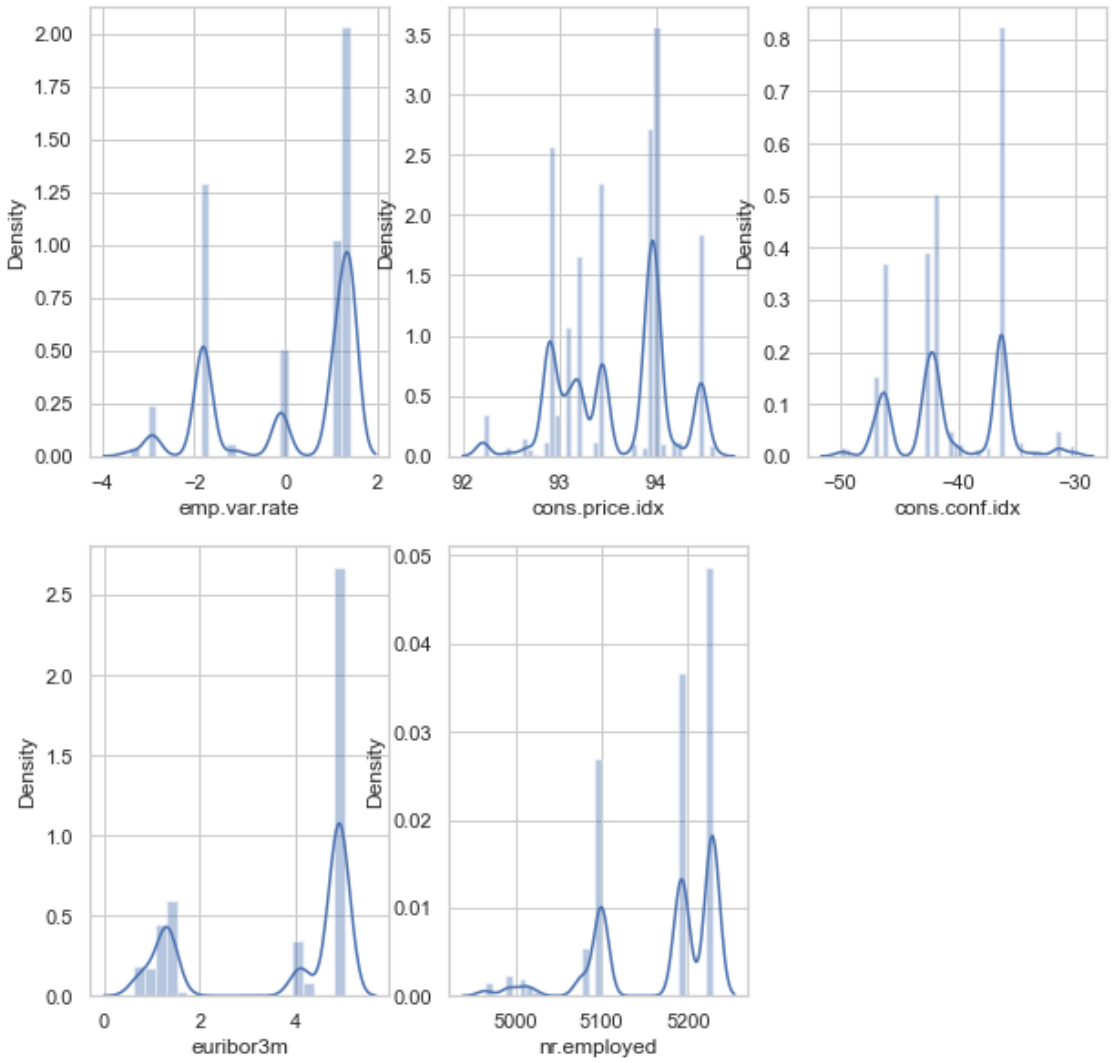
[40]:
```
#Biểu đồ thể hiện ma trận tương quan giữa các biến
fig, ax = plt.subplots(figsize=(15,9))
sns.heatmap(bank.corr(), vmin=-1, vmax=1, cmap=sns.diverging_palette(40, 440,␣
 ↪as_cmap=True), annot=True, linewidths=.5)
plt.show()
```



[41]:
```
#Vì pdays và các tính năng trước đó có tương quan với nhau nên chuyển đổi tính
↪năng trước đó thành tính năng phân loại.
bank.previous = bank.previous.apply(lambda x: 1 if x > 0 else 0).astype('uint8')
```

[42]:
```
#TRỰC QUAN HÓA DỮ LIỆU

plt.subplot(231)
sns.distplot(bank['emp.var.rate'])
fig = plt.gcf()
fig.set_size_inches(10,10)

plt.subplot(232)
sns.distplot(bank['cons.price.idx'])
fig = plt.gcf()
fig.set_size_inches(10,10)

plt.subplot(233)
sns.distplot(bank['cons.conf.idx'])
fig = plt.gcf()
fig.set_size_inches(10,10)

plt.subplot(234)
sns.distplot(bank['euribor3m'])
```

```
fig = plt.gcf()
fig.set_size_inches(10,10)

plt.subplot(235)
sns.distplot(bank['nr.employed'])
fig = plt.gcf()
fig.set_size_inches(10,10)


## Ta thấy tỉ lệ thay đổi nhân viên cao, nghĩa là ngân hàng đã thực hiện chiến␣
 ↪dịch khi có sự thay đổi cao do biến động kinh tế

## Chỉ số giá tiêu dùng cũng tốt cho thấy người dân đang có một mức giá hợp lý␣
 ↪để thanh toán các khoản chi tiêu hằng ngày.
##Vì vậy, mọi ngườii sẽ dư tiền và dễ sẽ nảy sinh ý tưởng tiết kiệm

## Chỉ số niềm tin giá thấp cho thấy người dân đang không có niềm tin vào nền␣
 ↪kinh tế biến động

## Lãi suất Euribor 3 tháng là lãi suất mà một số ngân hàng Châu Âu cho vay lẫn␣
 ↪nhau các khoản tiền bằng đồng Euro có thời hạn
## 3 tháng. Nhìn vào biểu đồ ta thấy, lãi suất này đang khá cao

##Số lượng nhân viên ngân hàng ở mức cao nhất ở mức cao nhất có thể làm tăng␣
 ↪chỉ số thu nhập. Vì vậy, ta có thể thực hiện
## chiến dịch marketing nhắm vào các nhân viên
```

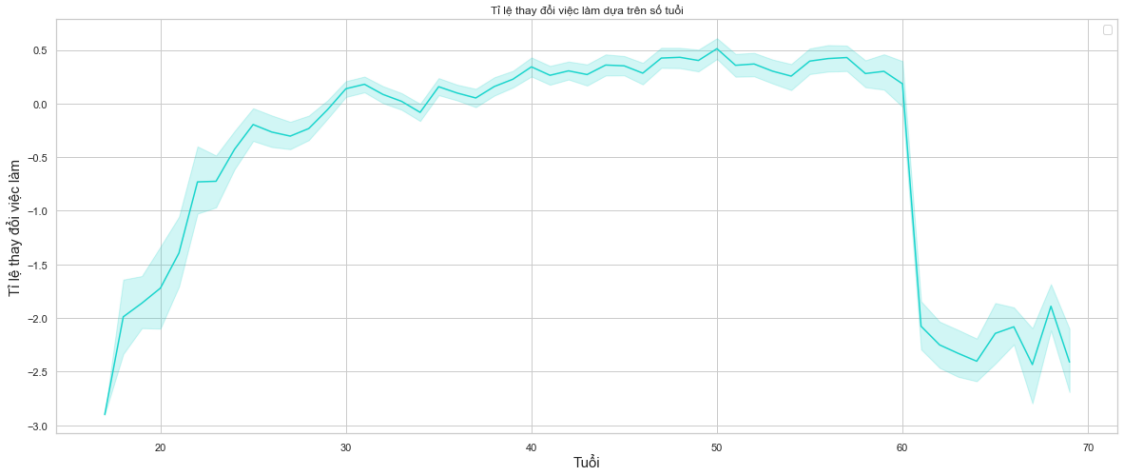


[43]:
```
import plotly.express as px

fig = px.box(bank, x="job", y="duration")
fig.update_traces(quartilemethod="exclusive") # or "inclusive", or "linear" by␣
↪default
fig.show()

##So sánh mức trung bình, giới văn phòng, doanh nhân có thời lượng cuộc gọi cao␣
↪và sinh viên,
##người đã nghỉ hưu có thời lượng trung bình ít hơn
```

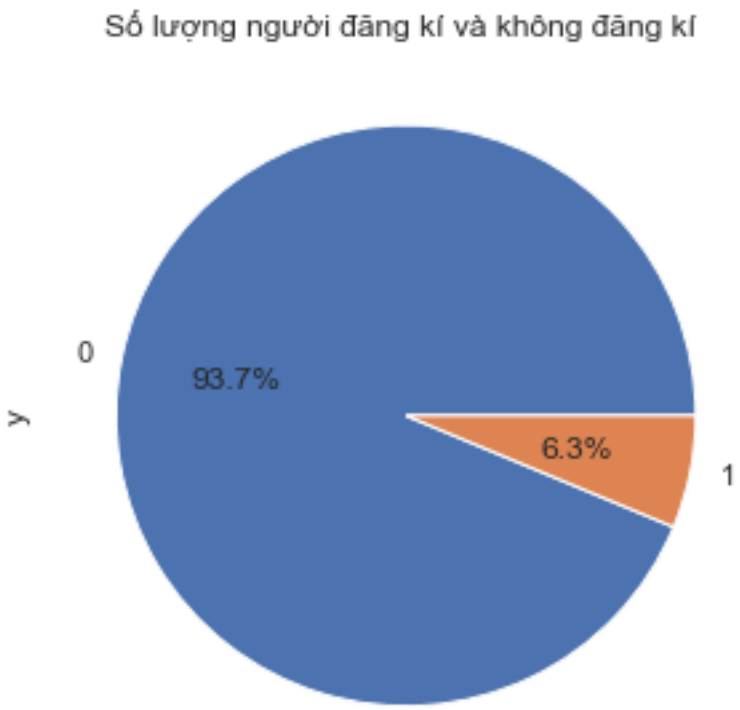
[44]:
```
plt.figure(figsize=(20,8))
sns.lineplot(data=bank,x='age',y='emp.var.rate',color='#1CD6CE')
plt.title('Tỉ lệ thay đổi việc làm dựa trên số tuổi',fontsize=12)
```

```
plt.legend(fontsize=12)
plt.xlabel('Tuổi',fontsize=15)
plt.ylabel('Tỉ lệ thay đổi việc làm',fontsize=15)
plt.show()


## Tỉ lệ biến động việc làm xảy ra đột ngột ở lứa tuổi từ 50 đến 60 tuổi. Vì␣
 ↪vậy, đây là độ tuổi nghỉ hưu người
#ta có xu hướng mở khoản gửi ngân hàng cần phải chú ý đẩy mạnh chiến dịch␣
 ↪marketing vào nhóm đối tượng này
##những đối tượng này. Ngoài ra tỉ lệ người có việc làm tăng dần mạnh từ 20 đến␣
 ↪30 tuổi, cần phải có những chính sách khuyến
##khích những đối tượng này vay
```

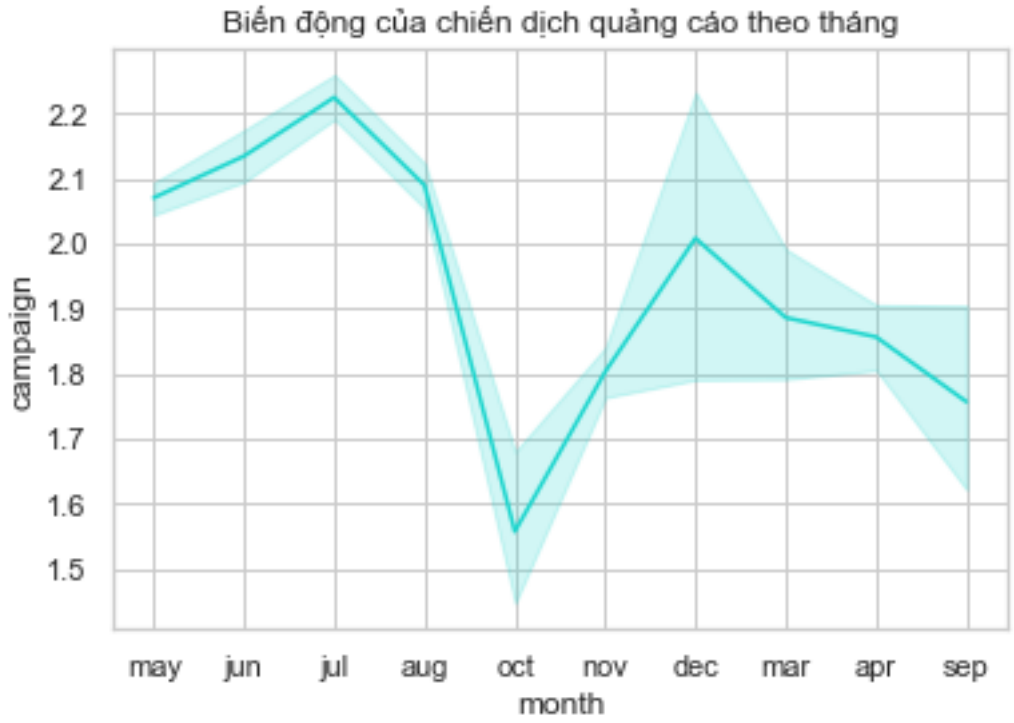
```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



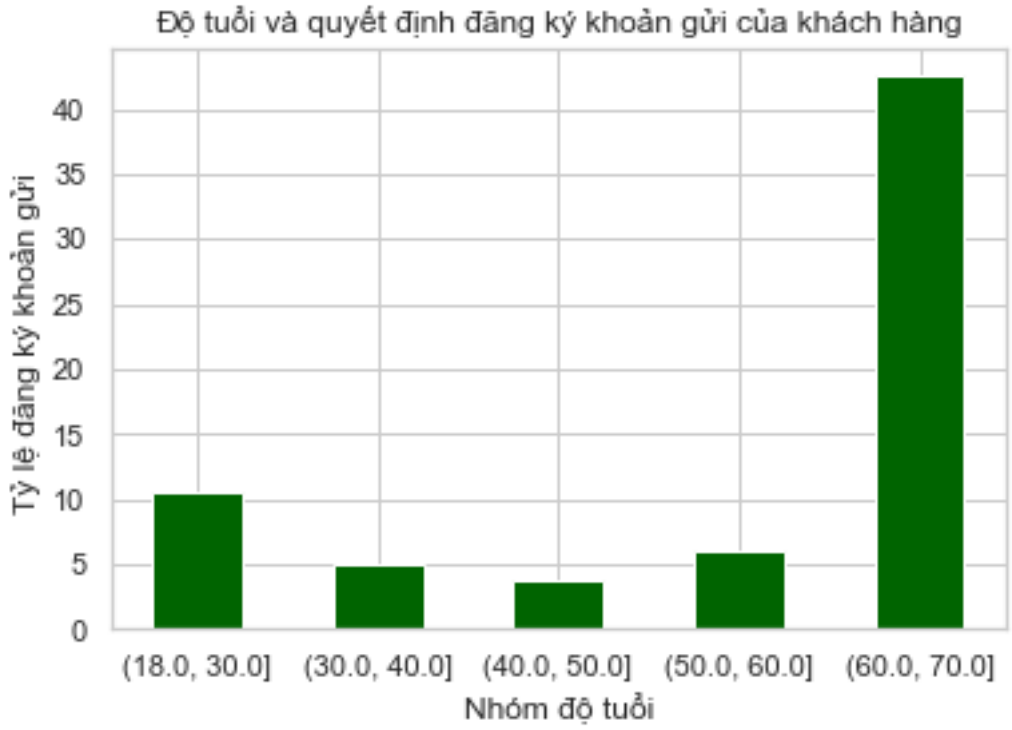
[45]:
```
bank['y'].value_counts().plot(kind='pie',figsize=(10,5),autopct='%1.1f%%')
plt.title('Số lượng người đăng kí và không đăng kí ')
plt.show()
##Nhìn chung, số lượng người không đăng kí chiếm phần lớn, gấp 15 lần so với␣
 ↪người đăng kí. Như vậy, những quảng cáo của
##ngân hàng đang không hiệu quả, cần được đổi mới cách tiếp cận để thu hút thêm␣
 ↪nhiều đối tượng hơn
```



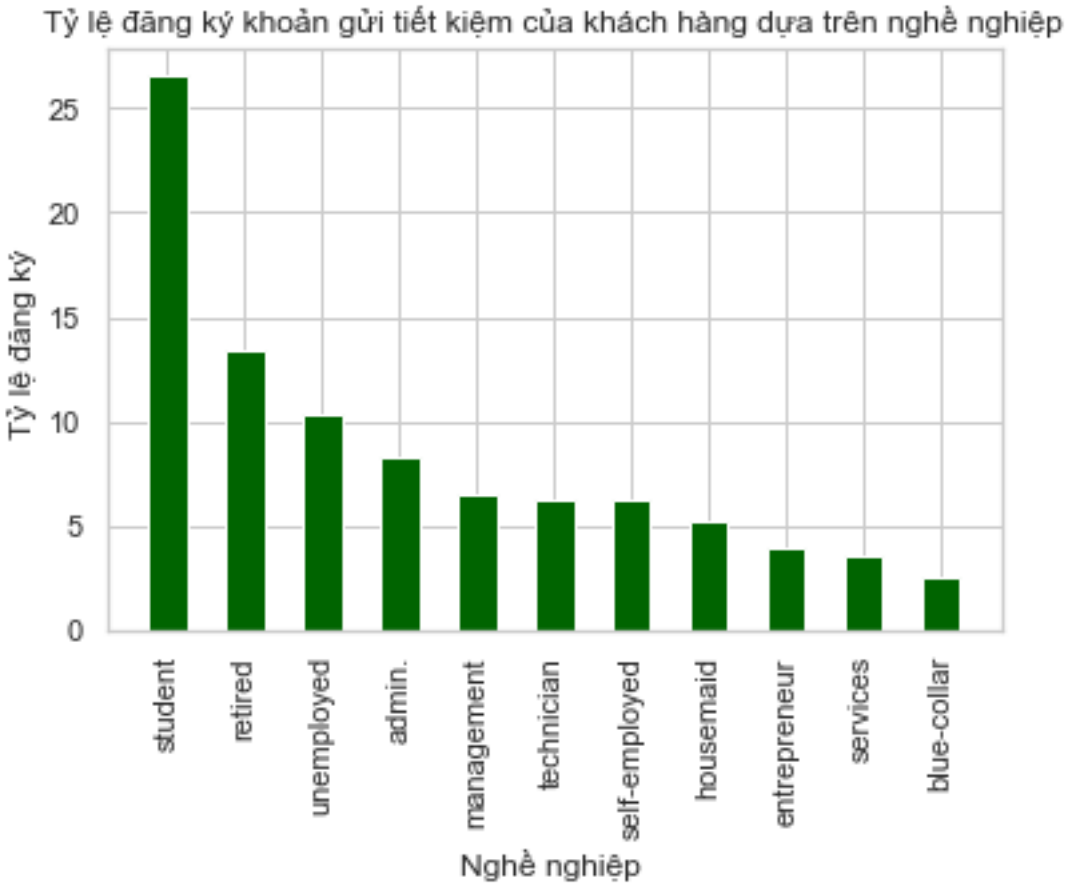
[46]:
```
plt.title('Biến động của chiến dịch quảng cáo theo tháng')
sns.lineplot(data=bank,x='month',y='campaign',color='#1CD6CE')
plt.show()

## Có thể thấy chiến dịch chủ yếu tập trung vào đầu kỳ ngân hàng (tháng 5, 6, 7)
##Thông thường giai đoạn giáo dục bắt đầu trong thời gian đó nên có khả năng␣
 ↪cha mẹ đặt cọc dưới tên của con cái họ
##ngân hàng cũng đã thực hiện chiến dịch của mình vào cuối kỳ ngân hàng.
```



[47]:
```
# Group clients into 6 age groups (18-30, 30-40, 40-50, 50-60, 60-70, >70)
conversionsAgeGroup = bank.groupby(pd.cut(bank['age'], bins=[18, 30, 40, 50,␣
 ↪60, 70, float('inf')]))
summary = conversionsAgeGroup['age'].count().reset_index(name='TotalCount')
summary['NumberConversions'] = conversionsAgeGroup['y'].sum().values
summary['ConversionRate'] = summary['NumberConversions'] /␣
 ↪summary['TotalCount'] * 100

# Rename the 6th group
summary['age'] = summary['age'].astype(str)
summary.loc[summary['age'] == 'inf', 'ageGroup'] = '70+'

# Visualizing conversions
plt.bar(summary['age'], summary['ConversionRate'], width=0.5, color='darkgreen')
plt.title('Độ tuổi và quyết định đăng ký khoản gửi của khách hàng')
plt.xlabel('Nhóm độ tuổi')
plt.ylabel('Tỷ lệ đăng ký khoản gửi ')
plt.show()
#Tỷ lệ đăng ký khoản gửi tiết kiệm của những người trên 60 đạt 40% trong số␣
 ↪những người cùng nhóm tuổi được tiếp thị
# Có thể kết luận, những người trên 60 tuổi phản ứng tốt hơn với chiến dịch␣
 ↪tiếp thị của ngân hàng so với các nhóm tuổi khác.
```



[48]:
```
# Group the data
conversionsJob = bank.groupby('job').agg(TotalCount=('job', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsJob['ConversionRate'] = conversionsJob['NumberConversions'] /␣
 ↪conversionsJob['TotalCount'] * 100
conversionsJob = conversionsJob.sort_values('ConversionRate', ascending=False)

# Visualizing conversions
plt.bar(conversionsJob['job'], conversionsJob['ConversionRate'], width=0.5,␣
 ↪color='darkgreen')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên nghề␣
 ↪nghiệp')
plt.xlabel('Nghề nghiệp ')
plt.ylabel('Tỷ lệ đăng ký ')
plt.xticks(rotation=90)
plt.show()
#Nhóm số 5 và 10, tức là Sinh viên và người đã về hưu có tỷ lệ đăng ký khoản␣
 ↪gửi cao hơn so với các nhóm "nghề nghiệp " khác
```



[49]:
```
# Group the data
conversionsEdu = bank.groupby('education').agg(TotalCount=('education',␣
 ↪'count'), NumberConversions=('y', 'sum')).reset_index()
conversionsEdu['ConversionRate'] = conversionsEdu['NumberConversions'] /␣
 ↪conversionsEdu['TotalCount'] * 100
conversionsEdu = conversionsEdu.sort_values('ConversionRate', ascending=False)

# Visualizing conversions by education
plt.bar(conversionsEdu['education'], conversionsEdu['ConversionRate'], width=0.
 ↪5, color='darkgreen')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên trình độ␣
 ↪giáo dục')
plt.xlabel('Trình độ giáo dục ')
plt.ylabel('Tỷ lệ đăng ký')
plt.xticks(rotation=90)
plt.show()
#Bằng đại học là nhóm có tỷ lệ đăng ký cao hơn mức trung bình
#bên cạnh đó nhóm mid.school hạn chế nỗ lực tiếp thị
```

[50]:
```
# Group the data
conversionsEdu = bank.groupby('marital').agg(TotalCount=('marital', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsEdu['ConversionRate'] = conversionsEdu['NumberConversions'] /␣
 ↪conversionsEdu['TotalCount'] * 100
conversionsEdu = conversionsEdu.sort_values('ConversionRate', ascending=False)



# Visualizing conversions
plt.bar(conversionsEdu['marital'], conversionsEdu['ConversionRate'], width=0.5,␣
 ↪color='darkgreen')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên tình trạng␣
 ↪hôn nhân')
plt.xlabel('Tình trạng hôn nhân ')
plt.ylabel('Tỷ lệ đăng ký')
plt.xticks(rotation=90)
```

```
plt.show()
#những người đã kết hôn có nhiều khả năng đăng ký khoản gửi tiết kiệm hơn
```

[51]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Group the data - housing loan
conversionsHousing = bank.groupby('housing').agg(TotalCount=('y', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsHousing['ConversionRate'] = conversionsHousing['NumberConversions'] /
 ↪ conversionsHousing['TotalCount'] * 100
conversionsHousing = conversionsHousing.sort_values('ConversionRate',␣
 ↪ascending=False)

# Visualizing the data - housing loan
plt.figure(figsize=(6, 4))
plt.bar(conversionsHousing['housing'], conversionsHousing['ConversionRate'],␣
 ↪width=0.5, color='darkgreen')
plt.xlabel('Khoản nợ vay để mua hoặc thuê nhà')
plt.ylabel('Tỷ lệ đăng ký ')
```

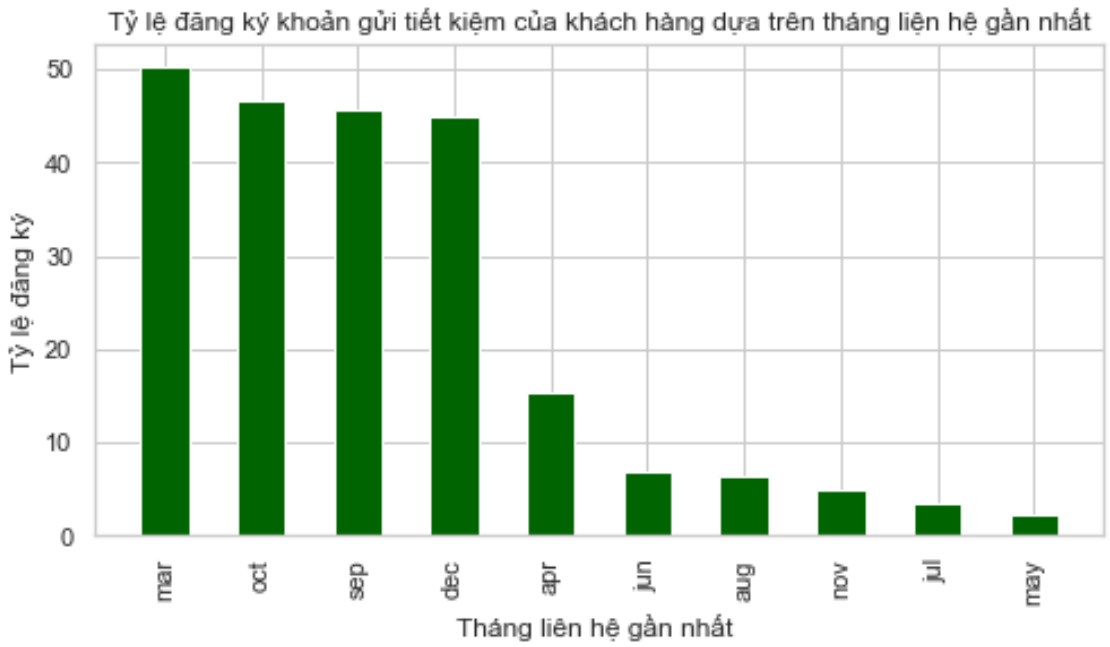
```
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên tình trạng␣
 ↪nợ vay hoặc thuê nhà')
plt.show()

#Vấn đề khách hàng có khoản vay mua nhà hay không nhìn chung không ảnh hưởng␣
 ↪đến quyết định đăng ký gửi tiết kiệm
#Vì tỷ lệ dựa trên biểu đồ không quá chêch lệch.
```

[52]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Group the data - personal loan
conversionsLoan = bank.groupby('loan').agg(TotalCount=('y', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsLoan['ConversionRate'] = conversionsLoan['NumberConversions'] /␣
 ↪conversionsLoan['TotalCount'] * 100
conversionsLoan = conversionsLoan.sort_values('ConversionRate', ascending=False)

# Visualizing the data - personal loan
plt.figure(figsize=(6, 4))
plt.bar(conversionsLoan['loan'], conversionsLoan['ConversionRate'], width=0.5,␣
 ↪color='darkgreen')
plt.xlabel('Khoản nợ cá nhân')
plt.ylabel('Tỷ lệ đăng ký ')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên tình trạng␣
 ↪nợ vay cá nhân')
```

```
plt.show()
#Những khách hàng không có khoản vay nợ cá nhân có tỷ lệ đăng ký tiết kiệm cao␣
 ↪hơn với những người đang chịu khoản vay này
#nhưng không quá chênh lệch.
```

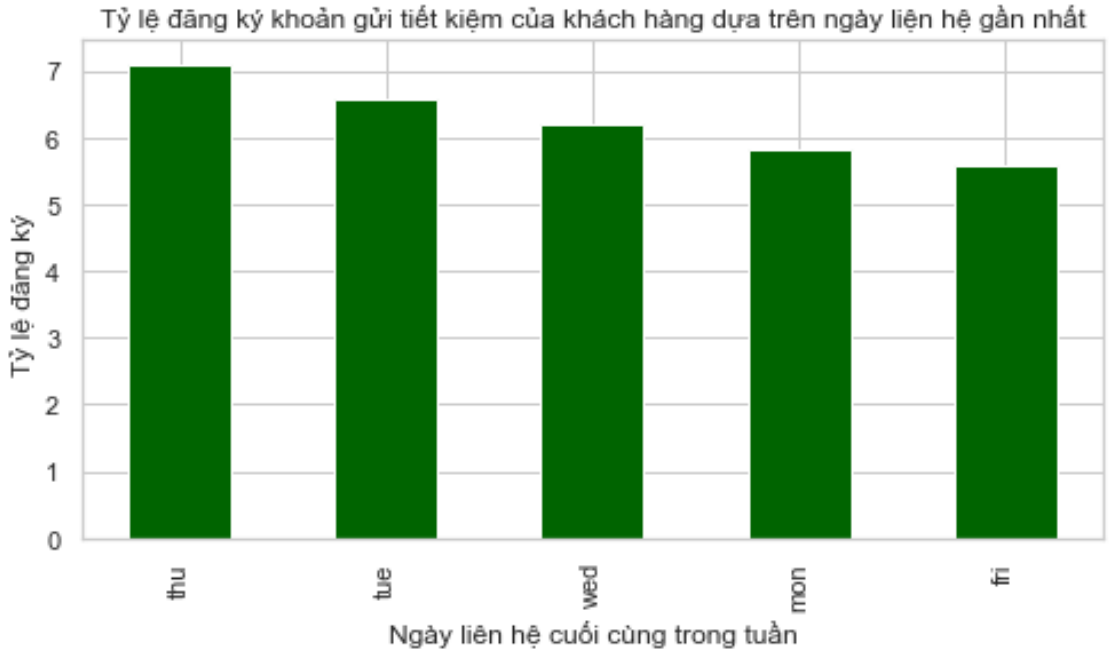
[53]:
```
# Group the data
conversionsLoan = bank.groupby('contact').agg(TotalCount=('y', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsLoan['ConversionRate'] = conversionsLoan['NumberConversions'] /␣
 ↪conversionsLoan['TotalCount'] * 100
conversionsLoan = conversionsLoan.sort_values('ConversionRate', ascending=False)

# Visualizing the data
plt.figure(figsize=(6, 4))
plt.bar(conversionsLoan['contact'], conversionsLoan['ConversionRate'], width=0.
 ↪5, color='darkgreen')
plt.xlabel('Phương thức liên hệ ')
plt.ylabel('Tỷ lệ đăng ký ')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng thông qua phương␣
 ↪thức liên hệ')
plt.show()
#Sự chêch lệch về kết quả chiến dịch marketing thông qua phương thức liên hệ␣
 ↪quá khác biệt'
#Có thể nói, liên hệ với khách hàng thông qua điện thoại di động hiệu quả hơn␣
 ↪so với điện thoại bàn.
```

[54]:
```
# Group the data by months
conversionsMonth = bank.groupby('month').agg(TotalCount=('y', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsMonth['ConversionRate'] = conversionsMonth['NumberConversions'] /␣
 ↪conversionsMonth['TotalCount'] * 100
conversionsMonth = conversionsMonth.sort_values('ConversionRate',␣
 ↪ascending=False)


# Visualizing the data
plt.figure(figsize=(8, 4))
plt.bar(conversionsMonth['month'], conversionsMonth['ConversionRate'], width=0.
 ↪5, color='darkgreen')
plt.xlabel('Tháng liên hệ gần nhất ')
plt.ylabel('Tỷ lệ đăng ký')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên tháng liện␣
 ↪hệ gần nhất')
plt.xticks(rotation=90)
plt.show()
#Những người được liên hệ lần cuối vào tháng 3, tháng 9, tháng 10 và tháng 12
#có tỷ lệ đăng ký khoản gửi tốt hơn nhiều so với nhóm còn lại.
```

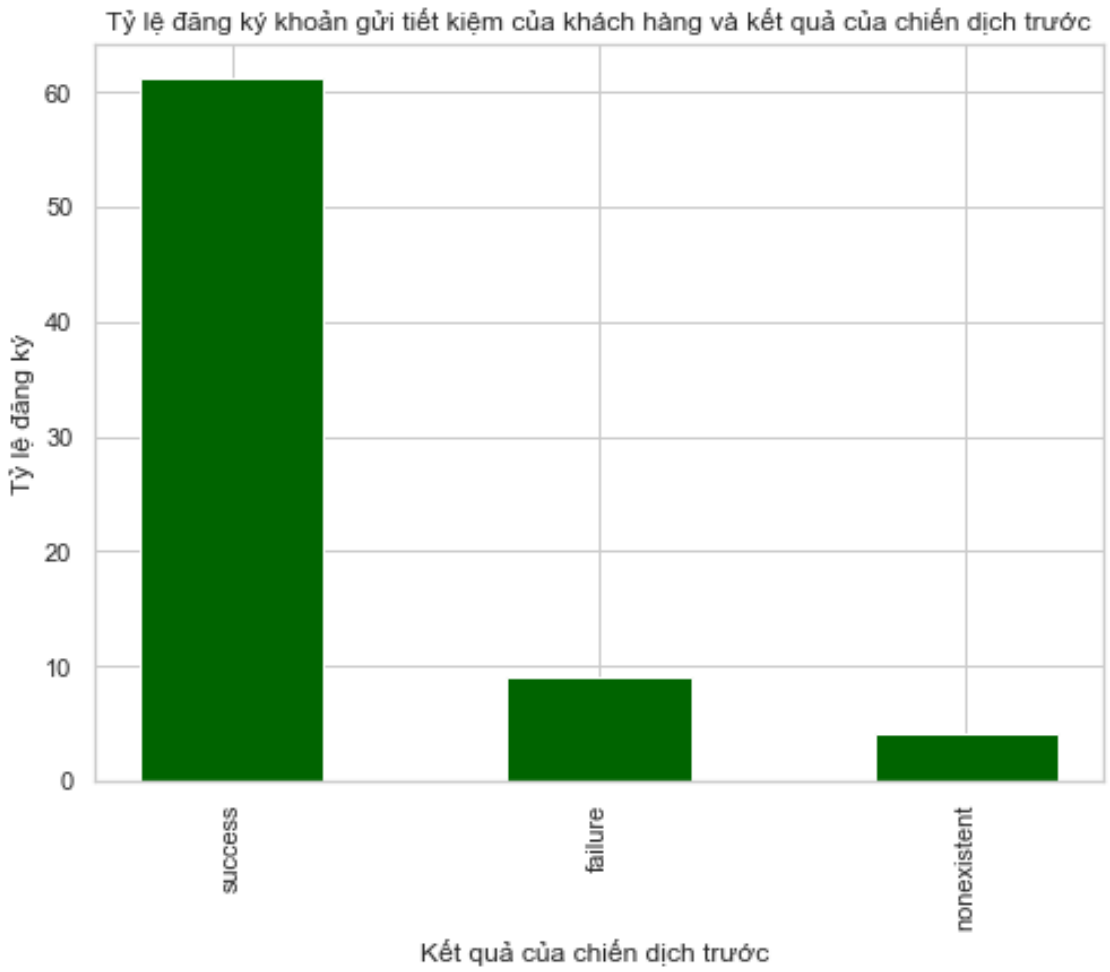


[55]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Group the data by days of the week
conversionsDayOfWeek = bank.groupby('day_of_week').agg(TotalCount=('y',␣
 ↪'count'), NumberConversions=('y', 'sum')).reset_index()
conversionsDayOfWeek['ConversionRate'] =␣
 ↪conversionsDayOfWeek['NumberConversions'] /␣
 ↪conversionsDayOfWeek['TotalCount'] * 100
conversionsDayOfWeek = conversionsDayOfWeek.sort_values('ConversionRate',␣
 ↪ascending=False)


# Visualizing the data
plt.figure(figsize=(8, 4))
plt.bar(conversionsDayOfWeek['day_of_week'],␣
 ↪conversionsDayOfWeek['ConversionRate'], width=0.5, color='darkgreen')
plt.xlabel('Ngày liên hệ cuối cùng trong tuần')
plt.ylabel('Tỷ lệ đăng ký ')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên ngày liện␣
 ↪hệ gần nhất')
plt.xticks(rotation=90)
plt.show()
#Tỷ lệ chuyển thành công của chiến dịch cao hơn nếu khách hàng được liên hệ vào␣
 ↪Thứ Năm, Thứ Ba và Thứ Tư.
```

```
#Tuy nhiên chênh lệch giữa các ngày trong tuần cũng không quá lớn.
```



[56]:
```
# Group the data by the previous outcome
conversionsPOutcome = bank.groupby('poutcome').agg(TotalCount=('y', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsPOutcome['ConversionRate'] =␣
 ↪conversionsPOutcome['NumberConversions'] / conversionsPOutcome['TotalCount']␣
 ↪* 100
conversionsPOutcome = conversionsPOutcome.sort_values('ConversionRate',␣
 ↪ascending=False)

# Visualizing the data
plt.figure(figsize=(8, 6))
plt.bar(conversionsPOutcome['poutcome'], conversionsPOutcome['ConversionRate'],␣
 ↪width=0.5, color='darkgreen')
plt.xlabel('Kết quả của chiến dịch trước ')
plt.ylabel('Tỷ lệ đăng ký')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng và kết quả của␣
 ↪chiến dịch trước')
plt.xticks(rotation=90)
plt.show()
#Rõ ràng, nếu kết quả của chiến dịch trước đó thành công thì kết quả của chiến␣
 ↪dịch lần này cũng khả quan hơn
#Những người quyết định đăng ký khoản gửi trong chiến dịch lần này có thể đã␣
 ↪trở thành khách hàng trung thành của ngân hàng.
```



[57]:
```
# Group the data
conversionsJob = bank.groupby('campaign').agg(TotalCount=('campaign', 'count'),␣
 ↪NumberConversions=('y', 'sum')).reset_index()
conversionsJob['ConversionRate'] = conversionsJob['NumberConversions'] /␣
 ↪conversionsJob['TotalCount'] * 100
conversionsJob = conversionsJob.sort_values('ConversionRate', ascending=False)

# Visualizing conversions
plt.bar(conversionsJob['campaign'], conversionsJob['ConversionRate'], width=0.
 ↪5, color='darkgreen')
plt.title('Tỷ lệ đăng ký khoản gửi tiết kiệm của khách hàng dựa trên số lần␣
 ↪liên hệ được thực hiện ')
plt.xlabel('Số lần liên hệ được thực hiện  ')
plt.ylabel('Tỷ lệ đăng ký ')
plt.xticks(rotation=90)
plt.show()
```

```
#Có thể thấy khi số lần liên hệ càng nhiều tỷ lệ đăng ký càng giảm
#Số lần liên hệ nhiều không đồng nghĩa với việc chiến dịch marketing tăng tỷ lệ␣
 ↪thành công
```

[58]:
```
#kiểm tra số biến object để Vectorise Features
{column: len(bank[column].unique()) for column in bank.select_dtypes('object').
 ↪columns}
```

[58]:
```
{'job': 11,
 'marital': 3,
 'education': 5,
 'housing': 2,
 'loan': 2,
 'contact': 2,
 'month': 10,
 'day_of_week': 5,
 'poutcome': 3}
```

[59]:
```
#Vectorise Features
bank['month']=bank['month'].replace({'may':5, 'jun':6, 'jul':7, 'aug':8 ,'oct':
 ↪10, 'nov':11 ,'dec':12, 'mar':3, 'apr':4,'sep':9})
```

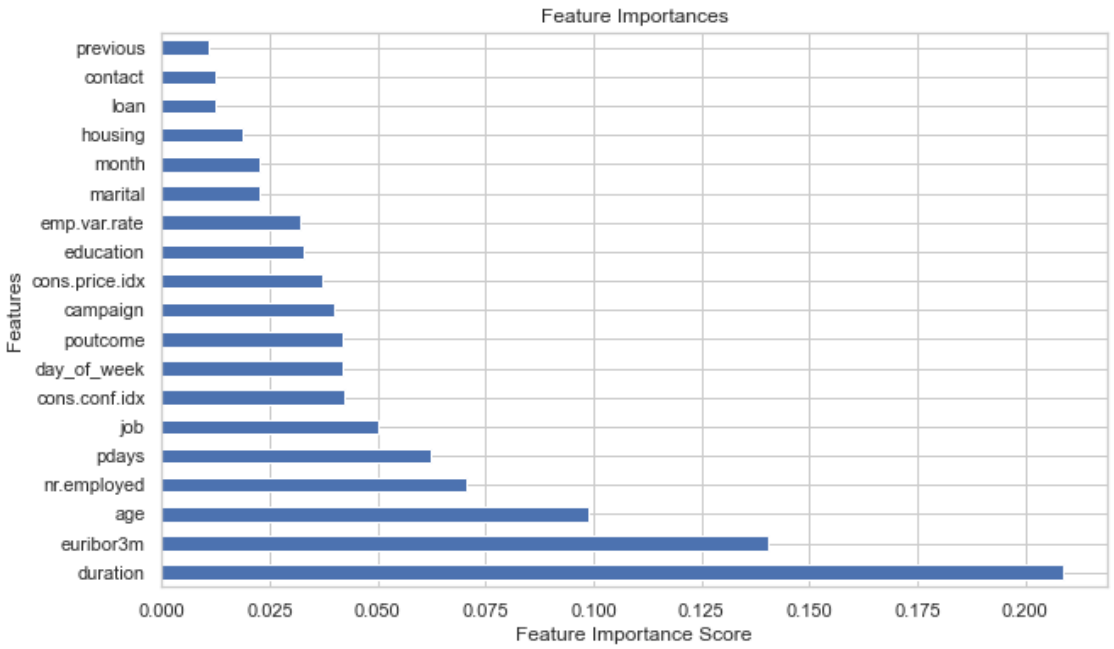
[60]:
```
bank['day_of_week']=bank['day_of_week'].replace({'mon':2, 'tue':3,'wed':4,'thu':
 ↪5, 'fri':6})
```

[61]:
```
le = preprocessing.LabelEncoder()
objects = ["job","marital","education","housing","loan","contact", "poutcome"]
for i in objects:
    bank[i] = le.fit_transform(bank[i])
```

[62]:
```
bank
```

[62]:
```
       age  job  marital  education  housing  loan  contact  month  \
0       56    3        1          2        0     0        1      5
1       57    7        1          0        0     0        1      5
2       37    7        1          0        1     0        1      5
3       40    0        1          2        0     0        1      5
4       56    7        1          0        0     1        1      5
…     …  …       …          …       …   …       …     …
41055   27    9        2          4        1     0        0     10
41056   60    5        1          3        0     1        0     10
41057   30    4        1          4        0     1        0     10
41058   65    5        1          2        1     0        0     10
41059   42    7        1          2        0     0        1     10

       day_of_week  duration  campaign  pdays  previous  poutcome  \
0                2       261         1      0         0         1
1                2       149         1      0         0         1
2                2       226         1      0         0         1
3                2       151         1      0         0         1
4                2       307         1      0         0         1
…              …       …       …       …    …       …
41055            4       184         1      0         1         0
41056            4       170         2      0         1         0
41057            4       255         1      6         1         2
41058            4       258         1      3         1         2
41059            5        71         1      0         0         1

       emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed  y
0               1.1          93.994          -36.4      4.857       5191.0  0
1               1.1          93.994          -36.4      4.857       5191.0  0
2               1.1          93.994          -36.4      4.857       5191.0  0
3               1.1          93.994          -36.4      4.857       5191.0  0
4               1.1          93.994          -36.4      4.857       5191.0  0
…             …              …               …          …           … ..
41055          -1.1          94.601          -49.5      1.043       4963.6  1
41056          -1.1          94.601          -49.5      1.043       4963.6  0
41057          -1.1          94.601          -49.5      1.043       4963.6  1
41058          -1.1          94.601          -49.5      1.043       4963.6  1
41059          -1.1          94.601          -49.5      1.045       4963.6  0

[32869 rows x 20 columns]
```

[63]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier

# Xác định đặc trưng (features) và biến mục tiêu (y)
X = bank.drop('y', axis=1)
y = bank['y']

# Huấn luyện mô hình RandomForestClassifier
model = RandomForestClassifier()
model.fit(X, y)

# Tạo Series chứa đặc trưng quan trọng
feat_importances = pd.Series(model.feature_importances_, index=X.columns)

# Vẽ biểu đồ thanh ngang của tất cả đặc trưng quan trọng
plt.figure(figsize=(10, 6))  # Kích thước biểu đồ
feat_importances.sort_values(ascending=False).plot(kind='barh')
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title('Feature Importances')
plt.show()
```



[64]:
```
#Feature Transformation
from scipy.stats import skew, kurtosis
feature_trans = Numerical_list.copy()
```

```
feature_normal = []
feature_non_normal = []

for i in feature_trans:
  skw = skew(bank[i])
  kts = kurtosis(bank[i])
  if skw < 2 and skw > -2:
    if kts < 2 and kts > -2:
      feature_normal.append(i)
    else:
      feature_non_normal.append(i)
  else:
    feature_non_normal.append(i)

print('Normal Distribusion:', feature_normal)
print('Non normal Distribusion:', feature_non_normal)
```

```
Normal Distribusion: ['age', 'duration', 'campaign', 'emp.var.rate',
'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']
Non normal Distribusion: ['pdays', 'previous']
```

[65]:
```
#drop đi những biến không quan trọng
feature = bank.columns.tolist()
del feature[19:]
feature.remove('contact')
feature.remove('previous')
feature.remove('housing')
feature.remove('loan')
feature
```

[65]:
```
['age',
 'job',
 'marital',
 'education',
 'month',
 'day_of_week',
 'duration',
 'campaign',
 'pdays',
 'poutcome',
 'emp.var.rate',
 'cons.price.idx',
 'cons.conf.idx',
 'euribor3m',
 'nr.employed']
```

[66]:
```
# Split Data into Train and Test
transform = feature_normal + feature_non_normal
x = bank[feature]
y = bank['y']
```

[67]:
```
x
```

[67]:

	age	job	marital	education	month	day_of_week	duration	campaign	\
0	56	3	1	2	5	2	261	1	
1	57	7	1	0	5	2	149	1	
2	37	7	1	0	5	2	226	1	
3	40	0	1	2	5	2	151	1	
4	56	7	1	0	5	2	307	1	
…	…	…	…	…	…	…	…	…	
41055	27	9	2	4	10	4	184	1	
41056	60	5	1	3	10	4	170	2	
41057	30	4	1	4	10	4	255	1	
41058	65	5	1	2	10	4	258	1	
41059	42	7	1	2	10	5	71	1	

	pdays	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	\
0	0	1	1.1	93.994	-36.4	
1	0	1	1.1	93.994	-36.4	
2	0	1	1.1	93.994	-36.4	
3	0	1	1.1	93.994	-36.4	
4	0	1	1.1	93.994	-36.4	
…	…	…	…	…	…	
41055	0	0	-1.1	94.601	-49.5	
41056	0	0	-1.1	94.601	-49.5	
41057	6	2	-1.1	94.601	-49.5	
41058	3	2	-1.1	94.601	-49.5	
41059	0	1	-1.1	94.601	-49.5	

	euribor3m	nr.employed
0	4.857	5191.0
1	4.857	5191.0
2	4.857	5191.0
3	4.857	5191.0
4	4.857	5191.0
…	…	…
41055	1.043	4963.6
41056	1.043	4963.6
41057	1.043	4963.6
41058	1.043	4963.6
41059	1.045	4963.6

[32869 rows x 15 columns]

[68]:
```
y
```

[68]:
```
0        0
1        0
2        0
3        0
4        0
        ..
41055    1
41056    0
41057    1
41058    1
41059    0
Name: y, Length: 32869, dtype: uint8
```

[69]:
```
from sklearn.model_selection import train_test_split
```

[70]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
↪random_state=3)
```

[71]:
```
bank['y_Class'] = bank['y']>0.9
print(bank['y_Class'].value_counts())
```

```
False    30810
True      2059
Name: y_Class, dtype: int64
```

[72]:
```
x = bank[[col for col in bank.columns if col not in ['y_Class','y']]].values
y = bank['y_Class'].values
print(x.shape)
print(y.shape)
```

```
(32869, 19)
(32869,)
```

[73]:
```
!pip install -U imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in d:\anacoda\lib\site-packages
(0.10.1)
Requirement already satisfied: joblib>=1.1.1 in d:\anacoda\lib\site-packages
(from imbalanced-learn) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in d:\anacoda\lib\site-packages
(from imbalanced-learn) (1.21.5)
Requirement already satisfied: scipy>=1.3.2 in d:\anacoda\lib\site-packages
(from imbalanced-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\anacoda\lib\site-
packages (from imbalanced-learn) (2.2.0)
```

```
Requirement already satisfied: scikit-learn>=1.0.2 in d:\anacoda\lib\site-
packages (from imbalanced-learn) (1.0.2)
```

[74]:
```
from imblearn import over_sampling

x_train_smote, y_train_smote = over_sampling.SMOTE(random_state=3).
 ↪fit_resample(x_train, y_train)
```

[75]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, cohen_kappa_score
from sklearn.model_selection import cross_validate
def eval_classification(model):
    y_pred = model.predict(x_test)
    y_pred_train = model.predict(x_train)
    y_pred_proba = model.predict_proba(x_test)
    y_pred_proba_train = model.predict_proba(x_train)

    print("Accuracy: ", round(accuracy_score(y_test, y_pred),3))
    print("Precision:",round(precision_score(y_test, y_pred),3))
    print("Recall: ",round(recall_score(y_test, y_pred),3))
    print("F1-Score: ",round(f1_score(y_test, y_pred),3))
    print("Kappa Score:", round(cohen_kappa_score(y_test, y_pred),3))
    score = cross_validate(model, x, y, cv=5, scoring='roc_auc',␣
 ↪return_train_score=True)
```

[76]:
```
# DecisionTree algorithm
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
model1 = DecisionTreeClassifier(random_state=3)
model1.fit(x_train, y_train)
eval_classification(model1)
```

```
Accuracy:  0.938
Precision: 0.473
Recall:  0.515
F1-Score:  0.493
Kappa Score: 0.46
```

[77]:
```
#  LightGBM algorithm
from lightgbm import LGBMClassifier
model2 = LGBMClassifier(random_state=3)
model2.fit(x_train, y_train)
eval_classification(model2)
```

```
Accuracy:  0.953
Precision: 0.624
Recall:  0.511
F1-Score:  0.562
```

```
Kappa Score: 0.538
```

[78]:
```python
#  RandomForest algorithm
from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestClassifier(random_state=3)
model3.fit(x_train, y_train)
y_pred = model3.predict(x_test)
eval_classification(model3)
```

```
Accuracy:  0.954
Precision: 0.646
Recall:  0.489
F1-Score:  0.557
Kappa Score: 0.533
```

[79]:
```python
# XGBoost algorithm
from xgboost import XGBClassifier
model4 = XGBClassifier(random_state=3)
model4.fit(x_train, y_train)
eval_classification(model4)
```

```
Accuracy:  0.955
Precision: 0.639
Recall:  0.547
F1-Score:  0.59
Kappa Score: 0.566
```

[80]:
```python
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, cohen_kappa_score

# Tạo danh sách các mô hình và tên tương ứng
models = [
    {'model': model1, 'name': 'model1'},
    {'model': model2, 'name': 'model2'},
    {'model': model3, 'name': 'model3'},
    {'model': model4, 'name': 'model4'}
]

# Tạo danh sách dictionaries để lưu kết quả đánh giá
results = []

# Duyệt qua danh sách các mô hình và tính toán kết quả đánh giá
for m in models:
    model = m['model']
    name = m['name']
    y_pred = model.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
```

```
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    kappa = cohen_kappa_score(y_test, y_pred)
    results.append({'Model': name, 'Accuracy':accuracy, 'Precision': precision,␣
 ↪'Recall': recall, 'F1-Score': f1, 'Kappa': kappa})

# Tạo DataFrame từ danh sách dictionaries và sắp xếp theo F1-Score giảm dần
df = pd.DataFrame(results)
df_sorted = df.sort_values(by='F1-Score', ascending=False)

# In DataFrame đã sắp xếp
print(df_sorted)
```

```
    Model  Accuracy  Precision    Recall  F1-Score     Kappa
3  model4  0.955278   0.639113  0.547496  0.589767  0.566267
1  model2  0.953250   0.624473  0.511226  0.562203  0.537769
2  model3  0.954264   0.646119  0.488774  0.556539  0.532916
0  model1  0.937836   0.473016  0.514680  0.492969  0.459920
```

[81]:
```
#  RandomForest algorithm  after Oversampling
from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestClassifier(random_state=3)
model3.fit(x_train_smote, y_train_smote)
y_pred = model3.predict(x_test)
eval_classification(model3)
```

```
Accuracy:  0.95
Precision: 0.56
Recall:  0.674
F1-Score:  0.611
Kappa Score: 0.585
```

[82]:
```
# XGBoost algorithm after Oversampling
from xgboost import XGBClassifier
model4 = XGBClassifier(random_state=3)
model4.fit(x_train_smote, y_train_smote)
eval_classification(model4)
```

```
Accuracy:  0.949
Precision: 0.552
Recall:  0.665
F1-Score:  0.603
Kappa Score: 0.576
```

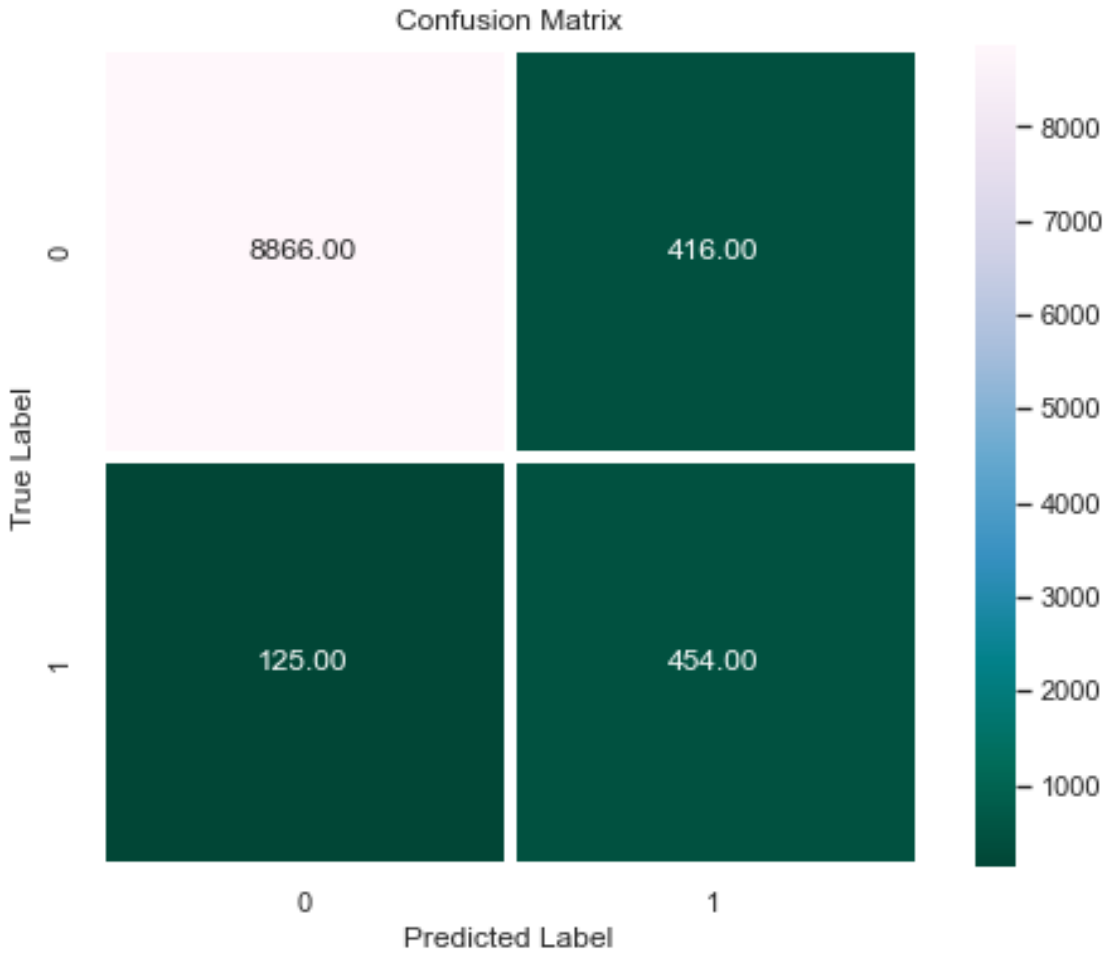
[83]:
```
#  LightGBM algorithm after Oversampling
from lightgbm import LGBMClassifier
```

```
model2 = LGBMClassifier(random_state=3)
model2.fit(x_train_smote, y_train_smote)
eval_classification(model2)
```

```
Accuracy:  0.945
Precision: 0.522
Recall:  0.784
F1-Score:  0.627
Kappa Score: 0.598
```

[84]:
```
#Mô hình LightGBM sẽ được chọn để dự báo khả năng đăng ký khoản gửi có kỳ hạn␣
↪của khách hàng
```

[85]:
```
# Dự đoán nhãn lớp cho dữ liệu kiểm tra
y_pred = model2.predict(x_test)

# Tính ma trận nhầm lẫn
matrix = confusion_matrix(y_test, y_pred)

# Vẽ biểu đồ ma trận nhầm lẫn
plt.figure(figsize=(8, 6))
sns.heatmap(matrix, annot=True, fmt=".2f", linewidths=5, square=True,␣
↪cmap="PuBuGn_r")
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



[86]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.96      0.97      9282
           1       0.52      0.78      0.63       579

    accuracy                           0.95      9861
   macro avg       0.75      0.87      0.80      9861
weighted avg       0.96      0.95      0.95      9861
```

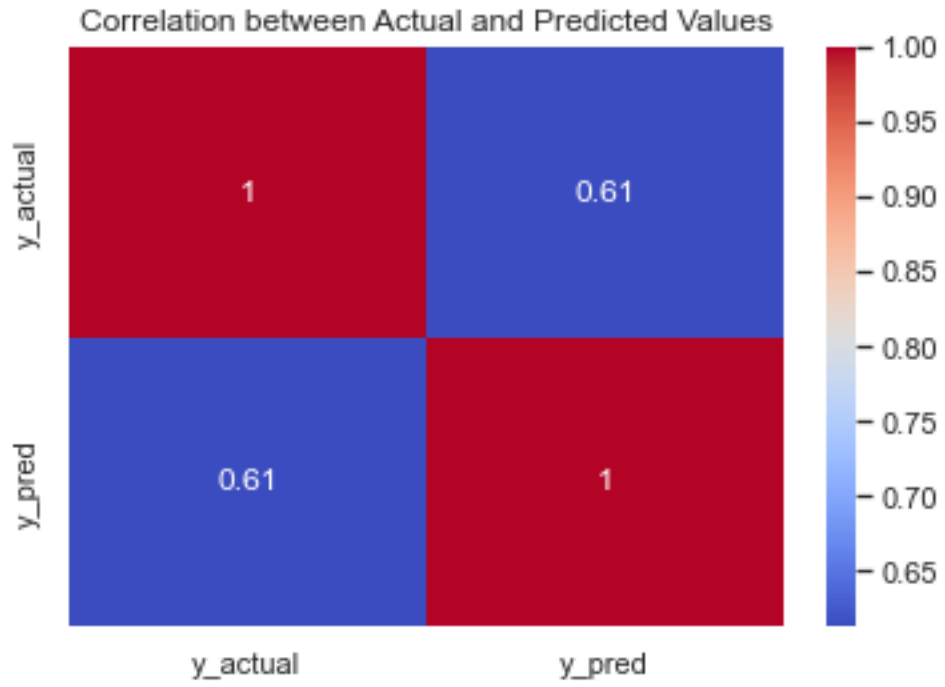
[87]:
```
#Trainig the model with
model2.fit(x_train_smote, y_train_smote)
```

```
# Predict the model with test data

y_pred = model2.predict(x_test)
out=pd.DataFrame({'y_actual':y_test,'y_pred':y_pred})
result=bank.merge(out,left_index=True,right_index=True)
result
```

[87]:

```
        age  job  marital  education  housing  loan  contact  month  \
5        45    7        1          2        0     0        1      5
11       25    7        2          0        1     0        1      5
13       57    3        0          2        1     0        1      5
22       55    5        2          0        1     0        1      5
23       41    9        2          0        1     0        1      5
…     …   …    …        …          …        …     …        …      …
41038    23    8        2          0        1     0        0     10
41045    30    8        2          0        1     0        0     10
41048    25    8        2          0        1     0        1     10
41056    60    5        1          3        0     1        0     10
41057    30    4        1          4        0     1        0     10

        day_of_week  duration  …  poutcome  emp.var.rate  cons.price.idx  \
5                 2       198  …         1           1.1          93.994
11                2       222  …         1           1.1          93.994
13                2       293  …         1           1.1          93.994
22                2       342  …         1           1.1          93.994
23                2       181  …         1           1.1          93.994
…               …       …  …  …       …            …              …
41038             2       226  …         0          -1.1          94.601
41045             3       196  …         0          -1.1          94.601
41048             3        17  …         1          -1.1          94.601
41056             4       170  …         0          -1.1          94.601
41057             4       255  …         2          -1.1          94.601

        cons.conf.idx  euribor3m  nr.employed  y  y_Class  y_actual  y_pred
5               -36.4      4.857       5191.0  0    False         0       0
11              -36.4      4.857       5191.0  0    False         0       0
13              -36.4      4.857       5191.0  0    False         0       0
22              -36.4      4.857       5191.0  0    False         0       0
23              -36.4      4.857       5191.0  0    False         0       0
…               …          …           …  ..      …        …     …
41038           -49.5      1.032       4963.6  1     True         1       1
41045           -49.5      1.037       4963.6  0    False         0       1
41048           -49.5      1.037       4963.6  0    False         0       0
41056           -49.5      1.043       4963.6  0    False         0       1
41057           -49.5      1.043       4963.6  1     True         1       1

[9861 rows x 23 columns]
```

[88]:
```
#Sử dụng biểu đồ heatmap với ma trận tương quan.
import seaborn as sns
import matplotlib.pyplot as plt

# Tính ma trận tương quan giữa 2 biến
corr_matrix = result[["y_actual", "y_pred"]].corr()

# Vẽ biểu đồ heatmap với ma trận tương quan
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation between Actual and Predicted Values")
plt.show()
```



[89]:
```
from sklearn.metrics import cohen_kappa_score

#có tập dữ liệu dự đoán từ mô hình (y_pred) và tập dữ liệu thực tế (y_true)
y_pred = model2.predict(x_test)
y_true = y_test

# Tính toán Cohen's Kappa
kappa = cohen_kappa_score(y_true, y_pred)

# In kết quả
print("Cohen's Kappa:", kappa)
```

```
Cohen's Kappa: 0.5983172063158185
```

[90]:
```
#Save model
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: