Importing libraries

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

```
df=pd.read_csv('/content/bank-additional-dataset.csv')
df
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutc
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	1	999	0	nonexis
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	1	999	0	nonexis
2	37	services	married	high.school	no	yes	no	telephone	may	mon	1	999	0	nonexis
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	1	999	0	nonexis
4	56	services	married	high.school	no	no	yes	telephone	may	mon	1	999	0	nonexis
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	1	999	0	nonexis
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	1	999	0	nonexis
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	2	999	0	nonexis
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	1	999	0	nonexis
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	3	999	1	fai

41188 rows × 20 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

EDA

```
#steps in EDA
#1.null values
#2.duplicate values
#3.outliers
#4.label encoding
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 20 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  campaign        41188 non-null  int64  
 11  pdays           41188 non-null  int64  
 12  previous        41188 non-null  int64  
 13  poutcome        41188 non-null  object 
 14  emp.var.rate    41188 non-null  float64
 15  cons.price.idx  41188 non-null  float64
 16  cons.conf.idx   41188 non-null  float64
 17  euribor3m       41188 non-null  float64
 18  nr.employed     41188 non-null  float64
 19  y               41188 non-null  object 
dtypes: float64(5), int64(4), object(11)
```

```
memory usage: 6.3+ MB
```

```
df.shape
```

```
(41188, 20)
```

```
df['y'].value_counts()
```

y	count
no	36548
yes	4640

dtype: int64

Handling null values

```
df.isnull().sum().sum()
#if you need to drop the null values:
#df.dropna(inplace=True)
#if you want to fill the null values:
#use fillna()
```

```
np.int64(0)
```

Handling Duplicate values

```
df.duplicated().sum()
```

```
np.int64(1784)
```

```
#To check the duplicate values
df[df.duplicated()==True]
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	pou
10	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	1	999	0	nonex
11	25	services	single	high.school	no	yes	no	telephone	may	mon	1	999	0	nonex
16	35	blue-collar	married	basic.6y	no	yes	no	telephone	may	mon	1	999	0	nonex
31	59	technician	married	unknown	no	yes	no	telephone	may	mon	1	999	0	nonex
104	52	admin.	divorced	university.degree	no	no	no	telephone	may	mon	1	999	0	nonex
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
39985	27	admin.	single	high.school	no	no	no	cellular	jun	tue	2	999	0	nonex
40401	31	student	single	unknown	no	yes	no	cellular	aug	thu	2	999	0	nonex
40404	41	entrepreneur	married	university.degree	no	yes	no	cellular	aug	thu	1	999	0	nonex
40806	35	technician	married	professional.course	no	yes	no	cellular	sep	thu	1	999	2	1
40840	32	admin.	single	university.degree	no	yes	no	cellular	sep	mon	4	999	0	nonex

1784 rows × 20 columns

```
#removing dupliacte values from dataset
df.drop_duplicates(inplace=True)
df.duplicated().sum()
```

```
np.int64(0)
```

```
#df after removing duplicates
df
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutc
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	1	999	0	nonexis
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	1	999	0	nonexis
2	37	services	married	high.school	no	yes	no	telephone	may	mon	1	999	0	nonexis
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	1	999	0	nonexis
4	56	services	married	high.school	no	no	yes	telephone	may	mon	1	999	0	nonexis
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	1	999	0	nonexis
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	1	999	0	nonexis
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	2	999	0	nonexis
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	1	999	0	nonexis
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	3	999	1	fai

39404 rows × 20 columns

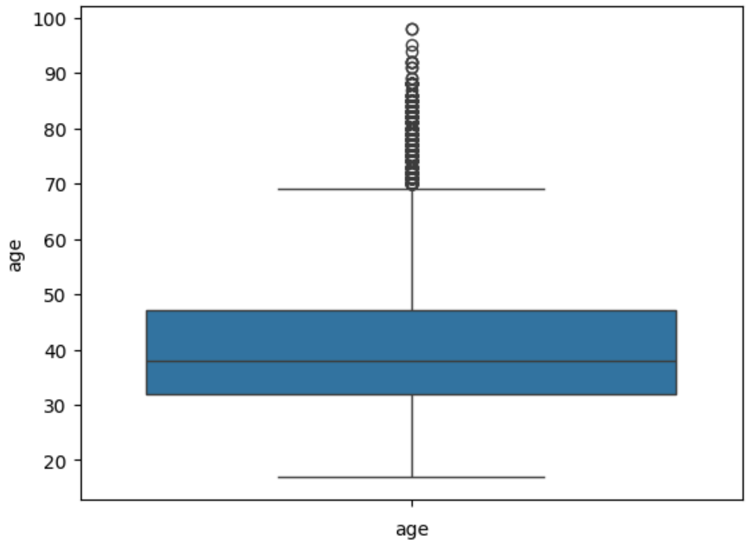
Next steps: Generate code with df | View recommended plots | New interactive sheet

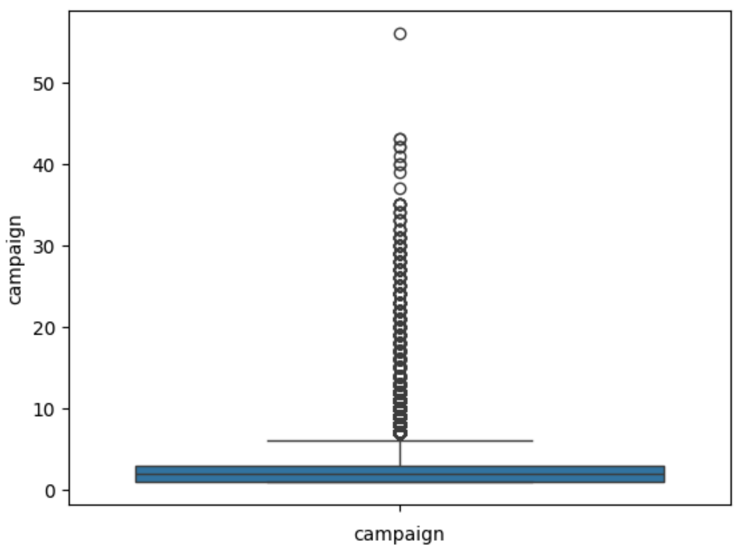
Outliers

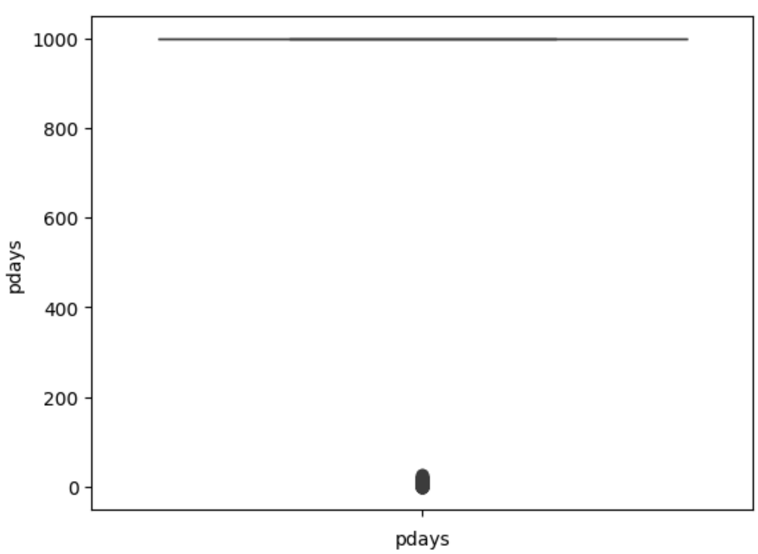
```
df.columns
```

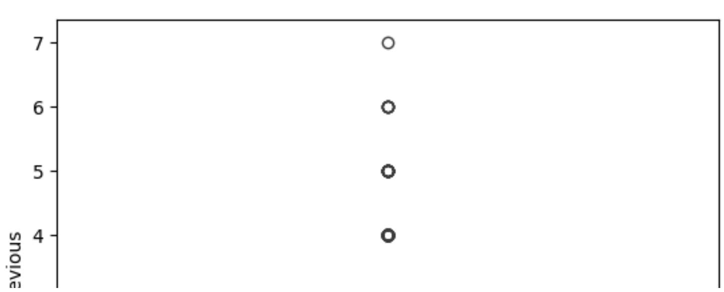
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'campaign', 'pdays', 'previous',
       'poutcome', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx',
       'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

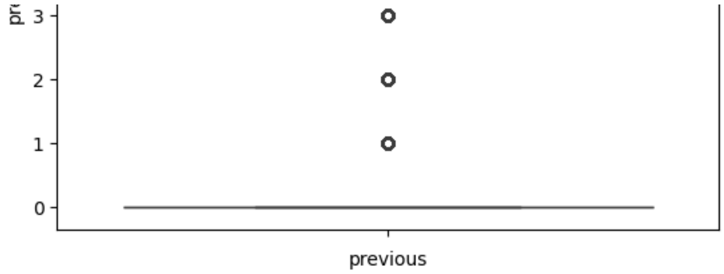
```
#outlier detection for all the columns in the dataset

for x in df.columns:
  if df[x].dtype=='int64' or df[x].dtype=='float64':
    sns.boxplot(df[x])
    plt.xlabel(x)
    plt.show()
```

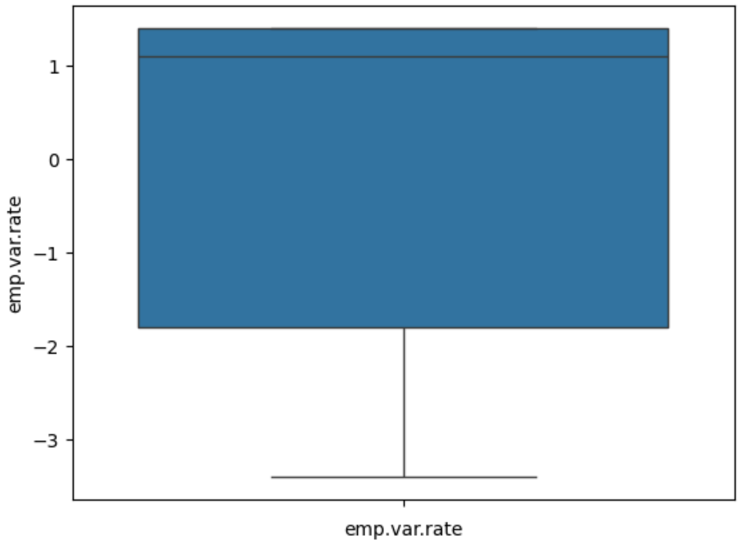

100
90
80
70
60
50
40
30
20
age
age

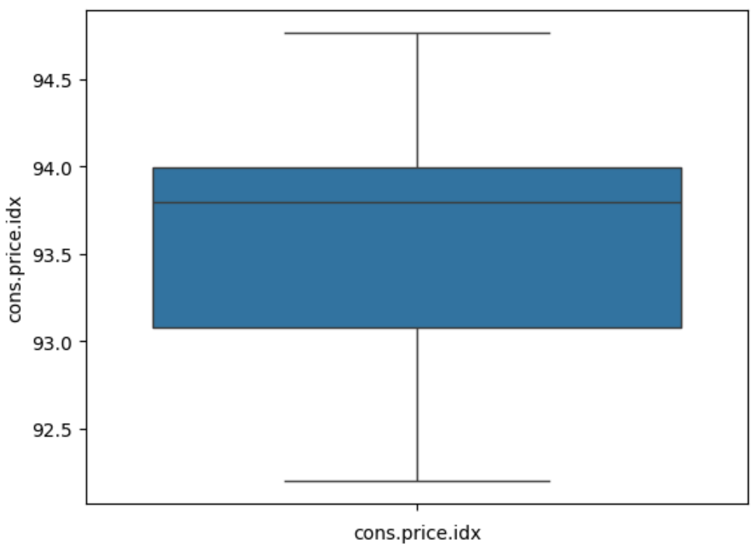

50
40
30
20
10
0
campaign
campaign

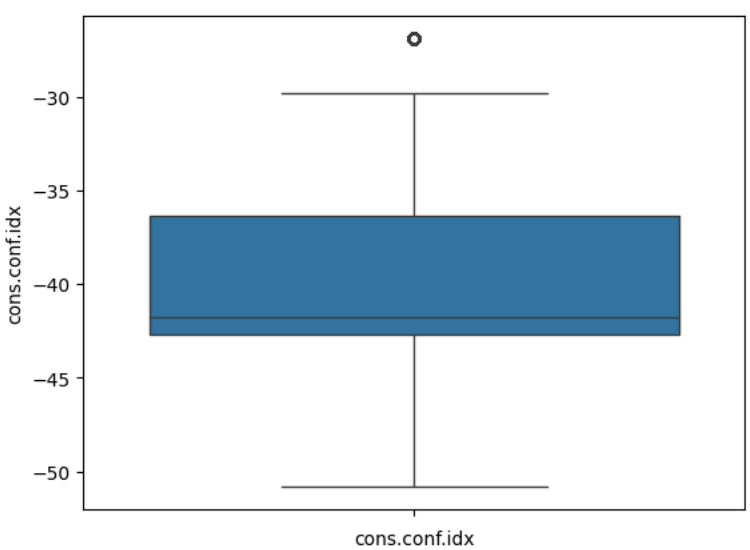

1000
800
600
400
200
0
pdays
pdays

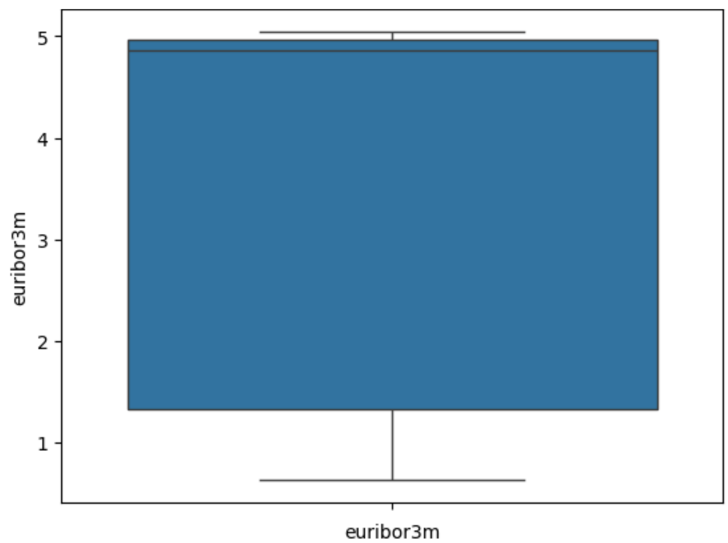

7
6
5
4

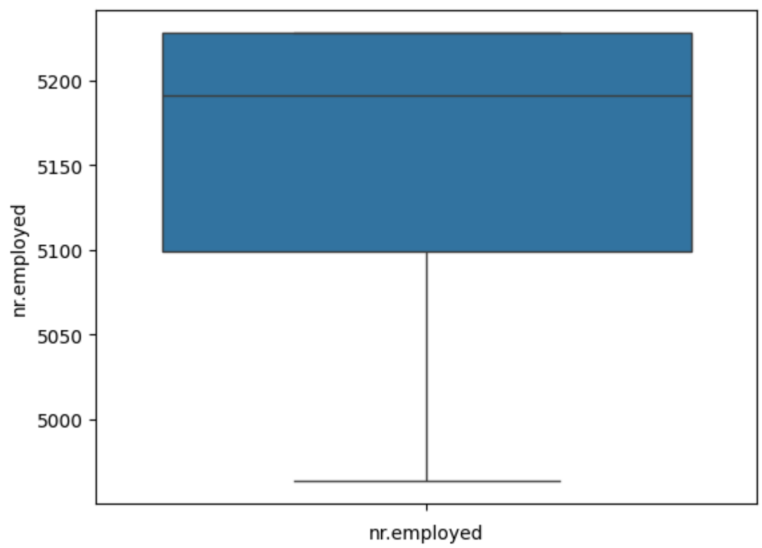

3
2
1
0
previous


1
0
−1
−2
−3
emp.var.rate
emp.var.rate


94.5
94.0
93.5
93.0
92.5
cons.price.idx
cons.price.idx


−30
−35
−40
−45
−50
cons.conf.idx
cons.conf.idx

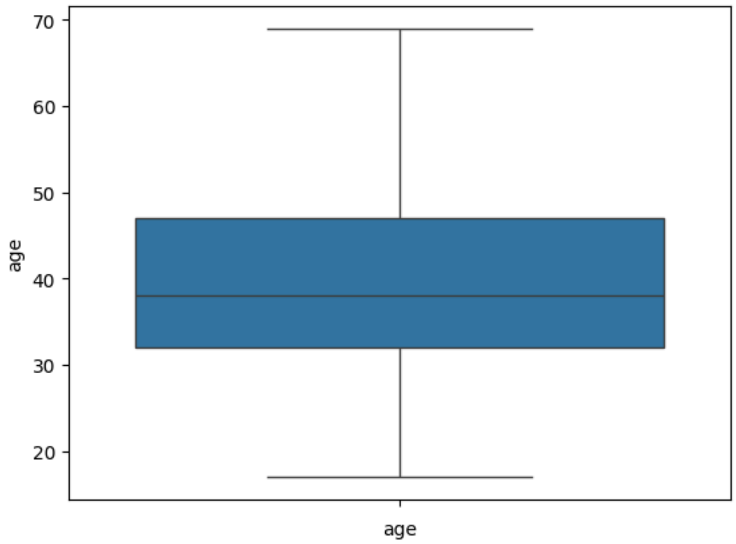

5
4
3
2
1
euribor3m
euribor3m

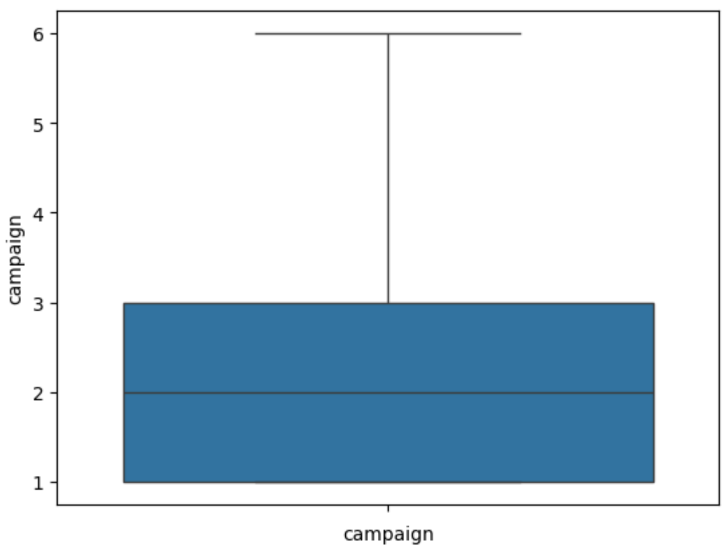

5200
5150
5100
5050
5000
nr.employed
nr.employed

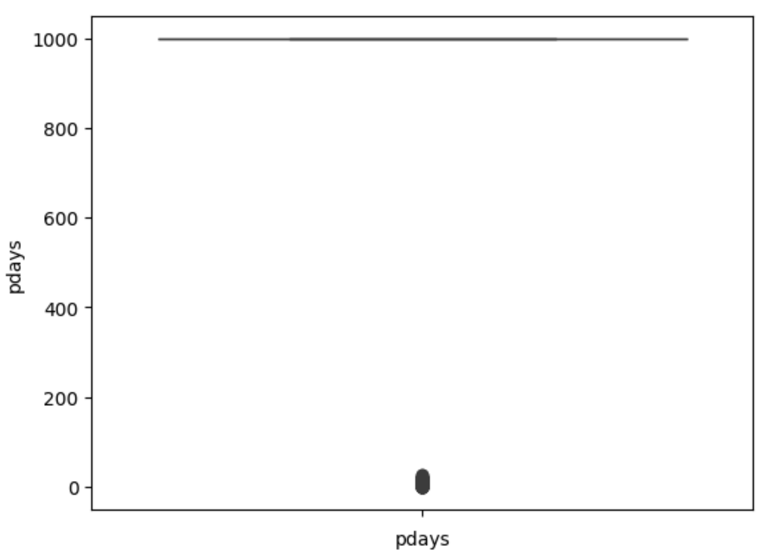
```
out_list=['age','campaign','cons.conf.idx']

for x in out_list:
  Q1=df[x].quantile(0.25)
  Q3=df[x].quantile(0.75)
  IQR=Q3-Q1
  lower_bound=Q1-1.5*IQR
  upper_bound=Q3+1.5*IQR
  print(f"lower_bound of {x}:",lower_bound)
  print(f"upper_bound of {x}:",upper_bound)
  df=df[(df[x]>=lower_bound) & (df[x]<=upper_bound)]
```

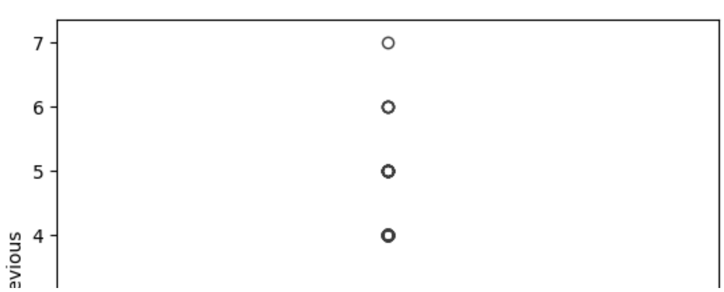
```
lower_bound of age: 9.5
upper_bound of age: 69.5
lower_bound of campaign: -2.0
upper_bound of campaign: 6.0
lower_bound of cons.conf.idx: -52.150000000000006
upper_bound of cons.conf.idx: -26.949999999999992
```

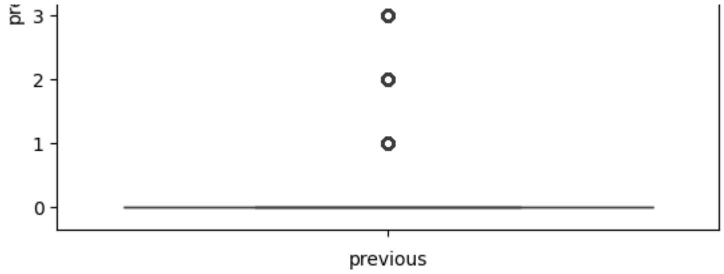
```
#vizual representation of the dataset after removing outliers

import matplotlib.pyplot as plt

for x in df.columns:
  if df[x].dtypes=='object' or x=='charges':
    continue
  else:
    sns.boxplot(df[x])
    plt.xlabel(x)
    plt.show()
```

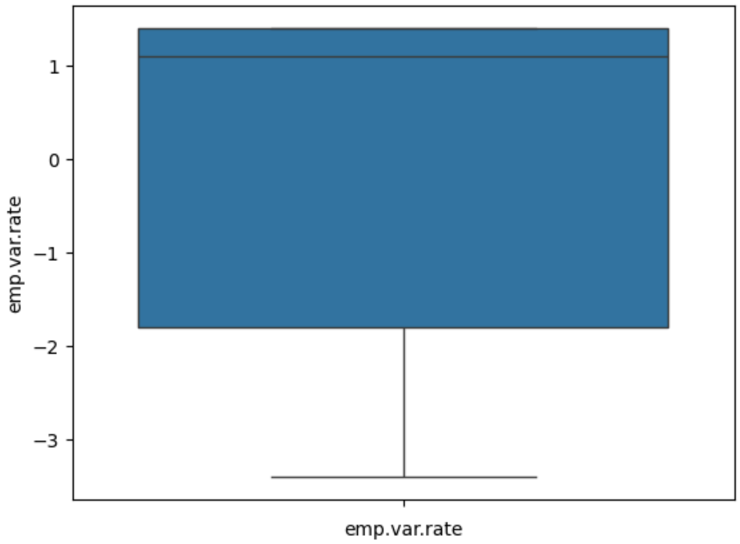

70
60
50
40
30
20
age
age

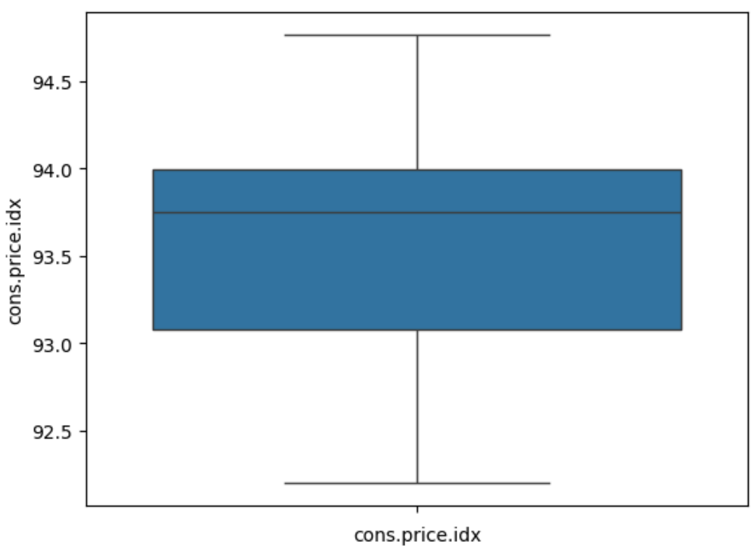

6
5
4
3
2
1
campaign
campaign

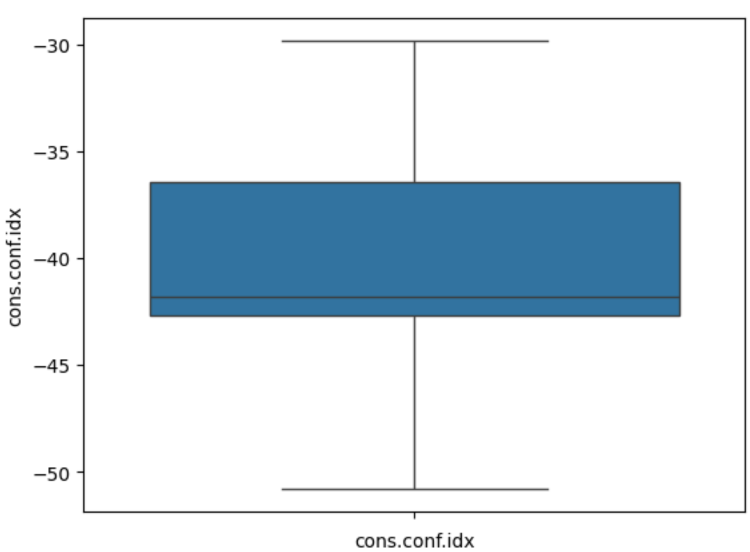

1000
800
600
400
200
0
pdays
pdays

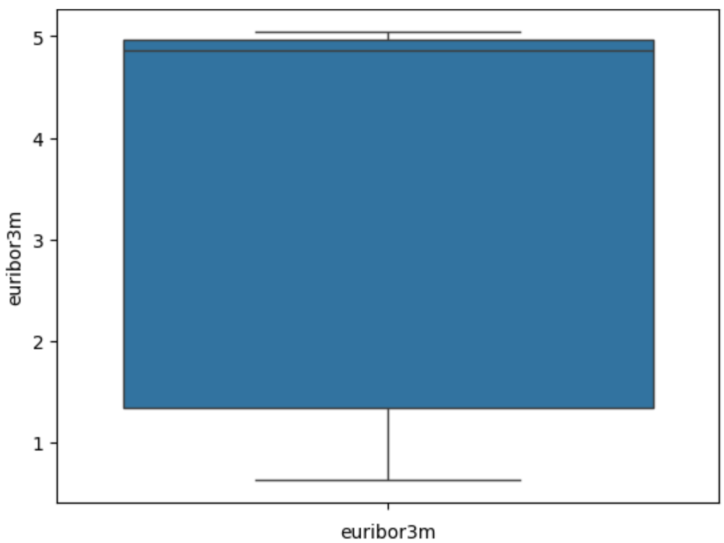

7
6
5
4


3
2
1
0
previous

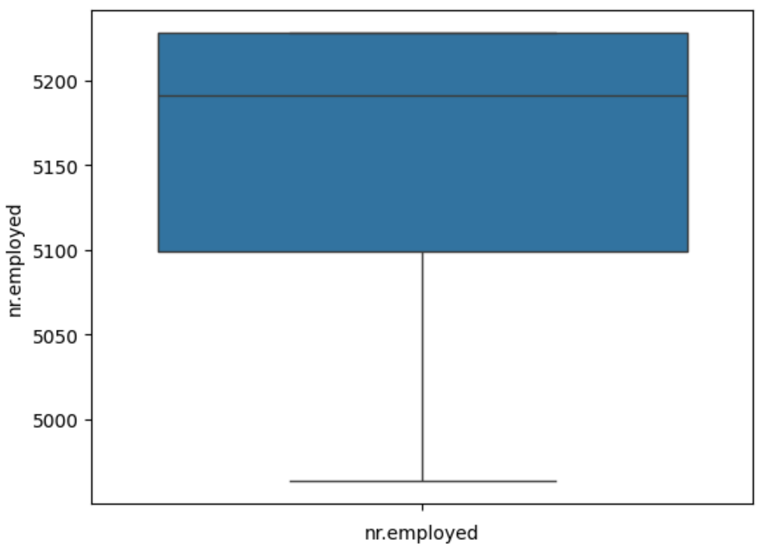

emp.var.rate
1
0
−1
−2
−3
emp.var.rate


cons.price.idx
94.5
94.0
93.5
93.0
92.5
cons.price.idx


cons.conf.idx
−30
−35
−40
−45
−50
cons.conf.idx



5
4
3
2
1
euribor3m
euribor3m



5200
5150
5100
5050
5000
nr.employed
nr.employed

Encoding

```
#Encoding - the process of converting categorical (non-numerical) data into a numerical format that machine learning algorithms can understa
#Common Encoding Techniques
# 1.One-Hot Encoding
# 2.Label Encoding
# 3.Ordinal Encoding
# 4.Mean Encoding
# 5.Frequency Encoding
```

```
#Label Encoding - Assigns a unique numerical label to each category
#We have to import Labelencoding sklearn.preprocessing
```

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

for x in df.columns:
  if df[x].dtype=='object':
    df[x]=le.fit_transform(df[x])

df
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rat
0	56	3	1	0	0	0	0	1	6	1	1	999	0	1	1
1	57	7	1	3	1	0	0	1	6	1	1	999	0	1	1
2	37	7	1	3	0	2	0	1	6	1	1	999	0	1	1
3	40	0	1	1	0	0	0	1	6	1	1	999	0	1	1
4	56	7	1	3	0	0	2	1	6	1	1	999	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41181	37	0	1	6	0	2	0	0	7	0	1	999	0	1	-1
41182	29	10	2	0	0	2	0	0	7	0	1	9	1	2	-1
41184	46	1	1	5	0	0	0	0	7	0	1	999	0	1	-1
41185	56	5	1	6	0	2	0	0	7	0	2	999	0	1	-1
41186	44	9	1	5	0	0	0	0	7	0	1	999	0	1	-1

36178 rows × 20 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 36178 entries, 0 to 41186
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             36178 non-null  int64  
 1   job             36178 non-null  int64  
 2   marital         36178 non-null  int64  
 3   education       36178 non-null  int64  
 4   default         36178 non-null  int64  
 5   housing         36178 non-null  int64  
 6   loan            36178 non-null  int64  
 7   contact         36178 non-null  int64  
 8   month           36178 non-null  int64  
 9   day_of_week     36178 non-null  int64  
 10  campaign        36178 non-null  int64  
 11  pdays           36178 non-null  int64  
 12  previous        36178 non-null  int64  
 13  poutcome        36178 non-null  int64  
 14  emp.var.rate    36178 non-null  float64
 15  cons.price.idx  36178 non-null  float64
 16  cons.conf.idx   36178 non-null  float64
 17  euribor3m       36178 non-null  float64
 18  nr.employed     36178 non-null  float64
 19  y               36178 non-null  int64  
dtypes: float64(5), int64(15)
```

```
memory usage: 5.8 MB
```

VIF

```
#VIF - Variation Inflation Factor
#The Variance Inflation Factor (VIF) is a measure of multicollinearity,
#which is a situation where multiple independent variables are highly correlated(depended) in a regression model
```

```
x=df.drop('y',axis=1)
y=df['y']
```

```
x.columns
```

```
Index(['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'],
      dtype='object')
```

```
#importing VIF

from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
#creating a datafrmame

vif_df=pd.DataFrame()
vif_df['features']=x.columns

vif_df
```

	features
0	age
1	job
2	marital
3	education
4	default
5	housing
6	loan
7	contact
8	month
9	day_of_week
10	campaign
11	pdays
12	previous
13	poutcome
14	emp.var.rate
15	cons.price.idx
16	cons.conf.idx
17	euribor3m
18	nr.employed

Next steps: Generate code with vif_df | View recommended plots | New interactive sheet

```
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)
```

```
vif_values
```

```
[np.float64(21.966070454402285),
 np.float64(2.1145506549496322),
 np.float64(5.649477190442339),
 np.float64(4.533743331984292),
 np.float64(1.413485603539284),
 np.float64(2.200633644198795),
 np.float64(1.2176088633103108),
 np.float64(2.935269716439509),
 np.float64(6.604836798011397),
 np.float64(3.0848606627114052),
 np.float64(3.665084550934581),
 np.float64(164.62831996241005),
 np.float64(6.028234902206734),
 np.float64(33.49721624624969),
 np.float64(37.49454522440181),
 np.float64(36695.73121886002),
 np.float64(143.30284378596946),
 np.float64(317.7508900878543),
 np.float64(41995.18904624725)]
```

```
vif_df['Multicollinearity']=vif_values
```

```
vif_df
```

	features	Multicollinearity
0	age	21.966070
1	job	2.114551
2	marital	5.649477
3	education	4.533743
4	default	1.413486
5	housing	2.200634
6	loan	1.217609
7	contact	2.935270
8	month	6.604837
9	day_of_week	3.084861
10	campaign	3.665085
11	pdays	164.628320
12	previous	6.028235
13	poutcome	33.497216
14	emp.var.rate	37.494545
15	cons.price.idx	36695.731219
16	cons.conf.idx	143.302844
17	euribor3m	317.750890
18	nr.employed	41995.189046

Next steps: Generate code with vif_df | View recommended plots | New interactive sheet

```
x.drop('nr.employed',axis=1,inplace=True)
x
```

age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rat
56	3	1	0	0	0	0	1	6	1	1	999	0	1	1.
57	7	1	3	1	0	0	1	6	1	1	999	0	1	1.
37	7	1	3	0	2	0	1	6	1	1	999	0	1	1.
40	0	1	1	0	0	0	1	6	1	1	999	0	1	1.
56	7	1	3	0	0	2	1	6	1	1	999	0	1	1.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
37	0	1	6	0	2	0	0	7	0	1	999	0	1	-1.
29	10	2	0	0	2	0	0	7	0	1	9	1	2	-1.
46	1	1	5	0	0	0	0	7	0	1	999	0	1	-1.
56	5	1	6	0	2	0	0	7	0	2	999	0	1	-1.
44	9	1	5	0	0	0	0	7	0	1	999	0	1	-1.

rows × 18 columns

Next steps: Generate code with x View recommended plots New interactive sheet

```
vif_df=pd.DataFrame()
vif_df['features']=x.columns
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)

vif_df['Multicollinearity']=vif_values
vif_df
```

	features	Multicollinearity
0	age	21.958629
1	job	2.114409
2	marital	5.648510
3	education	4.531664
4	default	1.410297
5	housing	2.200178
6	loan	1.217517
7	contact	2.468746
8	month	5.711382
9	day_of_week	3.084226
10	campaign	3.648469
11	pdays	164.584086
12	previous	5.936672
13	poutcome	33.257493
14	emp.var.rate	21.649652
15	cons.price.idx	699.184907
16	cons.conf.idx	123.353574
17	euribor3m	124.949516

Next steps: Generate code with vif_df View recommended plots New interactive sheet

```
x.drop('cons.price.idx',axis=1,inplace=True)
x
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rat
0	56	3	1	0	0	0	0	1	6	1	1	999	0	1	1
1	57	7	1	3	1	0	0	1	6	1	1	999	0	1	1
2	37	7	1	3	0	2	0	1	6	1	1	999	0	1	1
3	40	0	1	1	0	0	0	1	6	1	1	999	0	1	1
4	56	7	1	3	0	0	2	1	6	1	1	999	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41181	37	0	1	6	0	2	0	0	7	0	1	999	0	1	-1
41182	29	10	2	0	0	2	0	0	7	0	1	9	1	2	-1
41184	46	1	1	5	0	0	0	0	7	0	1	999	0	1	-1
41185	56	5	1	6	0	2	0	0	7	0	2	999	0	1	-1
41186	44	9	1	5	0	0	0	0	7	0	1	999	0	1	-1

36178 rows × 17 columns

Next steps: Generate code with x View recommended plots New interactive sheet

```
vif_df=pd.DataFrame()
vif_df['features']=x.columns
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)

vif_df['Multicollinearity']=vif_values
vif_df
```

	features	Multicollinearity
0	age	20.768073
1	job	2.108771
2	marital	5.496358
3	education	4.476751
4	default	1.410286
5	housing	2.198755
6	loan	1.217080
7	contact	2.371250
8	month	5.669544
9	day_of_week	3.066477
10	campaign	3.632626
11	pdays	91.086477
12	previous	3.056252
13	poutcome	16.894910
14	emp.var.rate	16.211514
15	cons.conf.idx	80.813154
16	euribor3m	89.550346

Next steps: Generate code with vif_df View recommended plots New interactive sheet

```
x.drop('pdays',axis=1,inplace=True)
x
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	previous	poutcome	emp.var.rate	con:
0	56	3	1	0	0	0	0	1	6	1	1	0	1	1.1	
1	57	7	1	3	1	0	0	1	6	1	1	0	1	1.1	
2	37	7	1	3	0	2	0	1	6	1	1	0	1	1.1	
3	40	0	1	1	0	0	0	1	6	1	1	0	1	1.1	
4	56	7	1	3	0	0	2	1	6	1	1	0	1	1.1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41181	37	0	1	6	0	2	0	0	7	0	1	0	1	-1.1	
41182	29	10	2	0	0	2	0	0	7	0	1	1	2	-1.1	
41184	46	1	1	5	0	0	0	0	7	0	1	0	1	-1.1	
41185	56	5	1	6	0	2	0	0	7	0	2	0	1	-1.1	
41186	44	9	1	5	0	0	0	0	7	0	1	0	1	-1.1	

36178 rows × 16 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

```
vif_df=pd.DataFrame()
vif_df['features']=x.columns
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)

vif_df['Multicollinearity']=vif_values
vif_df
```

	features	Multicollinearity
0	age	19.894134
1	job	2.105691
2	marital	5.379182
3	education	4.426096
4	default	1.410241
5	housing	2.195607
6	loan	1.216909
7	contact	2.301345
8	month	5.625194
9	day_of_week	3.053601
10	campaign	3.611721
11	previous	1.488630
12	poutcome	7.774289
13	emp.var.rate	12.646406
14	cons.conf.idx	41.645750
15	euribor3m	61.866669

Next steps: Generate code with vif_df | View recommended plots | New interactive sheet

```
x.drop('euribor3m',axis=1,inplace=True)
x
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	previous	poutcome	emp.var.rate	con
0	56	3	1	0	0	0	0	1	6	1	1	0	1	1.1	
1	57	7	1	3	1	0	0	1	6	1	1	0	1	1.1	
2	37	7	1	3	0	2	0	1	6	1	1	0	1	1.1	
3	40	0	1	1	0	0	0	1	6	1	1	0	1	1.1	
4	56	7	1	3	0	0	2	1	6	1	1	0	1	1.1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41181	37	0	1	6	0	2	0	0	7	0	1	0	1	-1.1	
41182	29	10	2	0	0	2	0	0	7	0	1	1	2	-1.1	
41184	46	1	1	5	0	0	0	0	7	0	1	0	1	-1.1	
41185	56	5	1	6	0	2	0	0	7	0	2	0	1	-1.1	
41186	44	9	1	5	0	0	0	0	7	0	1	0	1	-1.1	

36178 rows × 15 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

```
vif_df=pd.DataFrame()
vif_df['features']=x.columns
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)

vif_df['Multicollinearity']=vif_values
vif_df
```

	features	Multicollinearity
0	age	16.983749
1	job	2.099141
2	marital	5.187809
3	education	4.277073
4	default	1.409908
5	housing	2.187525
6	loan	1.216483
7	contact	2.289988
8	month	5.248013
9	day_of_week	3.012393
10	campaign	3.604623
11	previous	1.470503
12	poutcome	7.655584
13	emp.var.rate	1.556598
14	cons.conf.idx	30.977819

Next steps: Generate code with vif_df | View recommended plots | New interactive sheet

```
x.drop('cons.conf.idx',axis=1,inplace=True)
x
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	previous	poutcome	emp.var.rate
0	56	3	1	0	0	0	0	1	6	1	1	0	1	1.1
1	57	7	1	3	1	0	0	1	6	1	1	0	1	1.1
2	37	7	1	3	0	2	0	1	6	1	1	0	1	1.1
3	40	0	1	1	0	0	0	1	6	1	1	0	1	1.1
4	56	7	1	3	0	0	2	1	6	1	1	0	1	1.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41181	37	0	1	6	0	2	0	0	7	0	1	0	1	-1.1
41182	29	10	2	0	0	2	0	0	7	0	1	1	2	-1.1
41184	46	1	1	5	0	0	0	0	7	0	1	0	1	-1.1
41185	56	5	1	6	0	2	0	0	7	0	2	0	1	-1.1
41186	44	9	1	5	0	0	0	0	7	0	1	0	1	-1.1

36178 rows × 14 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

```
vif_df=pd.DataFrame()
vif_df['features']=x.columns
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)

vif_df['Multicollinearity']=vif_values
vif_df
```

	features	Multicollinearity
0	age	10.519901
1	job	2.089816
2	marital	4.300933
3	education	4.177050
4	default	1.409069
5	housing	2.152500
6	loan	1.212707
7	contact	2.268380
8	month	4.850603
9	day_of_week	2.943193
10	campaign	3.429230
11	previous	1.467364
12	poutcome	7.311764
13	emp.var.rate	1.540276

Next steps: Generate code with vif_df | View recommended plots | New interactive sheet

```
x.drop('age',axis=1,inplace=True)
x
```

```
x.drop('age',axis=1,inplace=True)
x
```

	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	previous	poutcome	emp.var.rate
0	3	1	0	0	0	0	1	6	1	1	0	1	1.1
1	7	1	3	1	0	0	1	6	1	1	0	1	1.1
2	7	1	3	0	2	0	1	6	1	1	0	1	1.1
3	0	1	1	0	0	0	1	6	1	1	0	1	1.1
4	7	1	3	0	0	2	1	6	1	1	0	1	1.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
41181	0	1	6	0	2	0	0	7	0	1	0	1	-1.1
41182	10	2	0	0	2	0	0	7	0	1	1	2	-1.1
41184	1	1	5	0	0	0	0	7	0	1	0	1	-1.1
41185	5	1	6	0	2	0	0	7	0	2	0	1	-1.1
41186	9	1	5	0	0	0	0	7	0	1	0	1	-1.1

36178 rows × 13 columns

```
vif_df=pd.DataFrame()
vif_df['features']=x.columns
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)

vif_df['Multicollinearity']=vif_values
vif_df
```

	features	Multicollinearity
0	job	2.065035
1	marital	4.291795
2	education	3.979094
3	default	1.322478
4	housing	2.097685
5	loan	1.209040
6	contact	2.266805
7	month	4.548502
8	day_of_week	2.842802
9	campaign	3.217133
10	previous	1.430428
11	poutcome	6.057195
12	emp.var.rate	1.540075

```
x.drop('poutcome',axis=1,inplace=True)
x
```

	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	previous	emp.var.rate
0	3	1	0	0	0	0	1	6	1	1	0	1.1
1	7	1	3	1	0	0	1	6	1	1	0	1.1
2	7	1	3	0	2	0	1	6	1	1	0	1.1
3	0	1	1	0	0	0	1	6	1	1	0	1.1
4	7	1	3	0	0	2	1	6	1	1	0	1.1

```
vif_df=pd.DataFrame()
vif_df['features']=x.columns
vif_values=[]
for i in range(len(x.columns)):
  vif=variance_inflation_factor(x.values,i)
  vif_values.append(vif)

vif_df['Multicollinearity']=vif_values
vif_df
```

36178 rows × 12 columns

	features	Multicollinearity
0	job	2.042861
1	marital	4.034238
2	education	3.721585
3	default	1.310552
4	housing	2.059338
5	loan	1.206251
6	contact	2.248569
7	month	4.390682
8	day_of_week	2.732719
9	campaign	3.071028
10	previous	1.376780
11	emp.var.rate	1.534253

Model Building

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

```
logistic_model=LogisticRegression()
```

```
logistic_model.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred=logistic_model.predict(x_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
y_test
```

	y
13733	0
20862	0
24835	0
15899	0
18937	0
...	...
38821	1
18239	0
2493	0
28102	0
22753	0

10854 rows × 1 columns

dtype: int64

Accuracy Score

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test,y_pred)*100        # Total_True_predictions % Total_no_of_predictions
```

```
88.48350838400589
```

```
from sklearn.metrics import confusion_matrix,classification_report
confusion_matrix(y_test,y_pred)
```

```
array([[9558,   25],
       [1225,   46]])
```