In [1]:
```python
# Algebra
import numpy as np

# DataFrame
import pandas as pd

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Algorithms
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import plot_confusion_matrix

from sklearn.model_selection import cross_validate
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier

import imblearn
import scipy
import math
```

Tasks to Perform :

Read in the file and get basic information about the data , including numerical summeries

In [2]:
```python
bdata = pd.read_csv(r'F:\ML\bank_marketing_data.csv')
```

In [3]:
```python
# NO of Rows and Columns in data :
bdata.shape
```

Out[3]:
```
(45211, 19)
```

In [4]:
```python
# Checkng the Presence of Null Values :
bdata.isnull().sum()
```

Out[4]:
```
age          0
job          0
salary       0
marital      0
education    0
targeted     0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
response     0
dtype: int64
```

- Observation : No null values present in data

In [5]:
```python
# Checking DataTypes in Data :
bdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 19 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   salary     45211 non-null  int64 
 3   marital    45211 non-null  object
 4   education  45211 non-null  object
 5   targeted   45211 non-null  object
 6   default    45211 non-null  object
 7   balance    45211 non-null  int64 
 8   housing    45211 non-null  object
 9   loan       45211 non-null  object
 10  contact    45211 non-null  object
 11  day        45211 non-null  int64 
 12  month      45211 non-null  object
 13  duration   45211 non-null  int64 
 14  campaign   45211 non-null  int64 
 15  pdays      45211 non-null  int64 
 16  previous   45211 non-null  int64 
 17  poutcome   45211 non-null  object
 18  response   45211 non-null  object
dtypes: int64(8), object(11)
memory usage: 6.6+ MB
```

- Observatioon : Two types of datatypes : int and object

In [6]:
```python
# Checking description of numerical class :
bdata.describe()
```

Out[6]:

	age	salary	balance	day	duration	campaign	pdays	
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45
mean	40.936210	57006.171065	1362.272058	15.806419	258.163080	2.763841	40.197828	
std	10.618762	32085.718415	3044.765829	8.322476	257.527812	3.098021	100.128746	
min	18.000000	0.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	
25%	33.000000	20000.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	
50%	39.000000	60000.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	
75%	48.000000	70000.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	
max	95.000000	120000.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	

- Observation :

1. Age of customer is Almost Normally Distributed

1. Salary of Customer is almost Normally Distributed

1. Account balance is skewed

In [7]:
```python
# Checking the Skewness of Numerical columns :
bdata.skew()
```

Out[7]:
```
age          0.684818
salary       0.137829
balance      8.360308
day          0.093079
duration     3.144318
campaign     4.898650
pdays        2.615715
previous    41.846454
dtype: float64
```

- Observation : Balance is Skewed as has been described previously
- Describe pdays column make note of the mean , median and minimum values . Anything fishy in the values ?
- ANS : Pdays Number of days that passed by after the client was last contacted from a previous compaign

In [8]:
```python
bdata['pdays'].describe()
```

Out[8]:
```
count    45211.000000
mean        40.197828
std        100.128746
min         -1.000000
25%         -1.000000
50%         -1.000000
75%         -1.000000
max        871.000000
Name: pdays, dtype: float64
```

In [9]:
```python
bdata['pdays'].skew()
```

Out[9]:
```
2.6157154736563477
```

- Mean = 40.2 days(approx)
- Median = -1 and minimum = -1
- Yes there is fishy values in the data i.e. (-1) , since no of days cann't be NEGATIVE

Q . Describe the pdays column again , this time limiting yourself to relevant values of pdays.How different are mean and median values ?

In [10]:
```python
bdata_1 = bdata.loc[bdata['pdays'] > -1]
```

In [11]:
```python
bdata_1['pdays'].describe()
```

Out[11]:
```
count    8257.000000
mean      224.577692
std       115.344035
min         1.000000
25%       133.000000
50%       194.000000
75%       327.000000
max       871.000000
Name: pdays, dtype: float64
```

In [12]:
```python
bdata_1['pdays'].skew()
```

Out[12]:
```
0.6931397093928039
```

- Observation : previously mean was 40 days and median was -1 day but now it changed to 224 days and 194 days respectively

Q . Plot a horizontal bar graph with the median values of balance for each education level value. Which group has the highest median?

In [13]:
```python
bdata.shape
```

Out[13]:
```
(45211, 19)
```

In [14]:
```python
bdata.head()
```

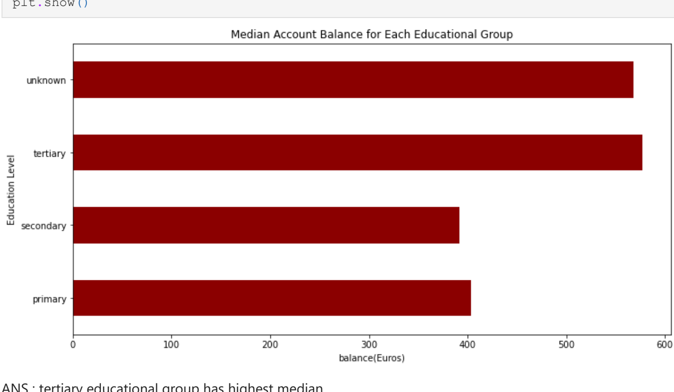
Out[14]:

	age	job	salary	marital	education	targeted	default	balance	housing	loan	contact	day
0	58	management	100000	married	tertiary	yes	no	2143	yes	no	unknown	5
1	44	technician	60000	single	secondary	yes	no	29	yes	no	unknown	5
2	33	entrepreneur	120000	married	secondary	yes	no	2	yes	yes	unknown	5
3	47	blue-collar	20000	married	unknown	no	no	1506	yes	no	unknown	5
4	33	unknown	0	single	unknown	no	no	1	no	no	unknown	5

In [15]:
```python
bdata['education'].unique()
```

Out[15]:
```
array(['tertiary', 'secondary', 'unknown', 'primary'], dtype=object)
```

In [195]:
```python
bdata_2 = bdata.groupby('education')['balance'].median().plot(kind = 'barh', figsize=
bdata_2.set_xlabel('balance(Euros)')
bdata_2.set_ylabel('Education Level')
bdata_2.set_title('Median Account Balance for Each Educational Group')
plt.show()
```

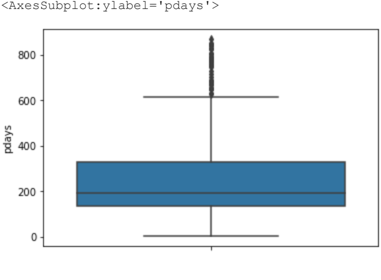


ANS : tertiary educational group has highest median

Q . Make a box plot for pdays. Do you see any outliers?

In [17]:
```python
sns.boxplot(y = 'pdays', data = bdata_1)
```

Out[17]:
```
<AxesSubplot:ylabel='pdays'>
```



- Checking outliers after removing -1 values

ANS : yes, there is presence of outlier (days greater than 600 are outliers)

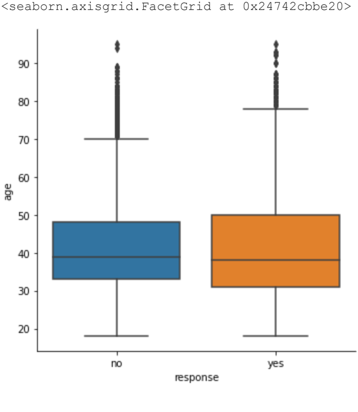
EDA Part :

TARGET VARIABLE : response

- effects of AGE group on response

In [18]:
```python
sns.catplot(y= 'age',x = 'response',kind = 'box', data = bdata)
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x24742cbbe20>
```



- OBSERVATION : THE age group doesn't have much effect on response of customer

EFFECTS OF JOB ON RESPONSE OF CUSTOMER

In [19]:
```python
bdata_3 = bdata.groupby('job')['response'].value_counts()
```

In [20]:
```python
print(bdata_3)
```

```
job            response
admin.         no          4540
               yes          631
blue-collar    no          9024
               yes          708
entrepreneur   no          1364
               yes          123
housemaid      no          1131
               yes          109
management     no          8157
               yes         1301
retired        no          1748
               yes          516
self-employed  no          1392
               yes          187
services       no          3785
               yes          369
student        no           669
               yes          269
technician     no          6757
               yes          840
unemployed     no          1101
               yes          202
unknown        no           254
               yes           34
Name: response, dtype: int64
```

In [21]:
```python
print(type(bdata_3))
```

```
<class 'pandas.core.series.Series'>
```

In [22]:
```python
bdata_3 = bdata_3.to_frame()
```

In [23]:
```python
print(bdata_3)
```

```
                        response
job           response          
admin.        no            4540
              yes            631
blue-collar   no            9024
              yes            708
entrepreneur  no            1364
              yes            123
housemaid     no            1131
              yes            109
management    no            8157
              yes           1301
retired       no            1748
              yes            516
self-employed no            1392
              yes            187
services      no            3785
              yes            369
student       no             669
              yes            269
technician    no            6757
              yes            840
unemployed    no            1101
              yes            202
unknown       no             254
              yes             34
```

In [24]:
```python
bdata_3.shape
```

Out[24]:
```
(24, 1)
```

In [25]:
```python
# admin
631/(631+ 4540)
```

Out[25]:
```
0.12202668729452718
```

In [26]:
```python
#blue-collar
708/(708+9024)
```

Out[26]:
```
0.07274969173859433
```

In [27]:
```python
# entrepreneur
123/(123+1364)
```

Out[27]:
```
0.08271687962340282
```

In [28]:
```python
# housemaid
109/(109+1131)
```

Out[28]:
```
0.08790322580645162
```

In [29]:
```python
# management
1301/(1301+8157)
```

Out[29]:
```
0.13755550856417847
```

In [30]:
```python
# retired
516/(516+1748)
```

Out[30]:
```
0.22791519434628976
```

In [31]:
```python
# self-employed
187/(187+1392)
```

Out[31]:
```
0.11842938568714376
```

In [32]:
```python
# services
369/(369+3785)
```

Out[32]:
```
0.08883004333172845
```

In [33]:
```python
# student
269/(269+669)
```

Out[33]:
```
0.2867803837953092
```

In [34]:
```python
# technician
840/(840+6757)
```

Out[34]:
```
0.11056996182703699
```

In [35]:
```python
# unemployed
202/(202+1101)
```

Out[35]:
```
0.15502686108979277
```

In [36]:
```python
# unknown
34/(34+254)
```

Out[36]:
```
0.11805555555555555
```

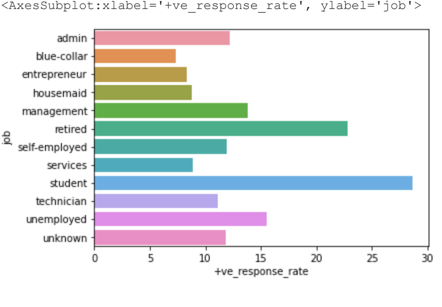
In [37]:
```python
data = {"job": ['admin', 'blue-collar', 'entrepreneur', 'housemaid',
        'management', 'retired', 'self-employed', 'services', 'student',
        'technician', 'unemployed', 'unknown'], "+ve_response_rate":[12.2,7.3,8.3,8.8,
df = pd.DataFrame(data)
```

In [38]:
```python
print(df)
```

```
              job  +ve_response_rate
0           admin               12.2
1     blue-collar                7.3
2    entrepreneur                8.3
3       housemaid                8.8
4      management               13.8
5         retired               22.8
6   self-employed               11.9
7        services                8.9
8         student               28.7
9      technician               11.1
10     unemployed               15.5
11        unknown               11.8
```

In [39]:
```python
sns.barplot(y= 'job', x = '+ve_response_rate', data = df)
```

Out[39]:
```
<AxesSubplot:xlabel='+ve_response_rate', ylabel='job'>
```

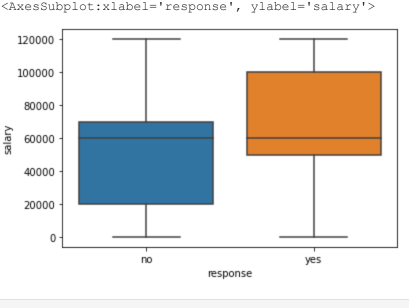


- Observation : student > retired > unemployed > management > admin

EFFECTS OF SALARY ON RESPONSE RATE

In [40]:
```python
sns.boxplot(y= 'salary', x= 'response', data = bdata)
```

Out[40]:
```
<AxesSubplot:xlabel='response', ylabel='salary'>
```

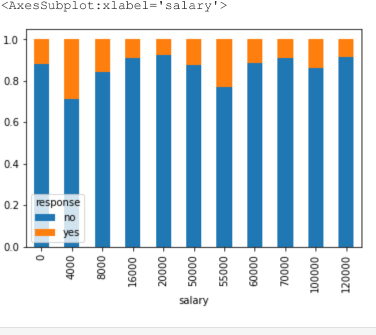


In [41]:
```python
bdata['salary'].unique()
```

Out[41]:
```
array([100000,  60000, 120000,  20000,      0,  55000,  50000,  70000,
         8000,  16000,   4000], dtype=int64)
```

In [42]:
```python
bdata.groupby('salary')['response'].value_counts(normalize = True).unstack('response')
```

Out[42]:
```
<AxesSubplot:xlabel='salary'>
```



In [43]:
```python
bdata.groupby('salary')['response'].value_counts()
```

Out[43]:
```
salary  response
0       no           254
        yes           34
4000    no           669
        yes          269
8000    no          1101
        yes          202
16000   no          1131
        yes          109
20000   no          9024
        yes          708
50000   no          4540
        yes          631
55000   no          1748
        yes          516
60000   no          8149
        yes         1027
70000   no          3785
        yes          369
100000  no          8157
        yes         1301
120000  no          1364
        yes          123
Name: response, dtype: int64
```

In [44]:
```python
109/(109+1131)
```

Out[44]:
```
0.08790322580645162
```

In [45]:
```python
708/(708+9024)
```

Out[45]:
```
0.07274969173859433
```

In [46]:
```python
1027/(1027+8149)
```

Out[46]:
```
0.11192240627724499
```

In [47]:
```python
369/(369+3785)
```

Out[47]:
```
0.08883004333172845
```

In [48]:
```python
1301/(1301+8157)
```

Out[48]:
```
0.13755550856417847
```

In [49]:
```python
123/(123+1364)
```

Out[49]:
```
0.08271687962340282
```

- OBSERVATION : the customer who has higher salary tend to give POSITIVE RESPONSE MORE

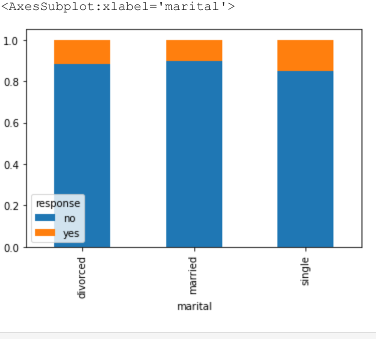
EFFECTS OF MARITAL STATUS ON RESPONSE RATE

In [50]:
```python
bdata['marital'].unique()
```

Out[50]:
```
array(['married', 'single', 'divorced'], dtype=object)
```

In [51]:
```python
bdata.groupby('marital')['response'].value_counts(normalize = True).unstack('response
```

Out[51]:
```
<AxesSubplot:xlabel='marital'>
```



In [52]:
```python
bdata.groupby('marital')['response'].value_counts()
```

Out[52]:
```
marital   response
divorced  no           4585
          yes           622
married   no          24459
          yes          2755
single    no          10878
          yes          1912
Name: response, dtype: int64
```

In [53]:
```python
622/(622+4585)
```

Out[53]:
```
0.11945458037257538
```

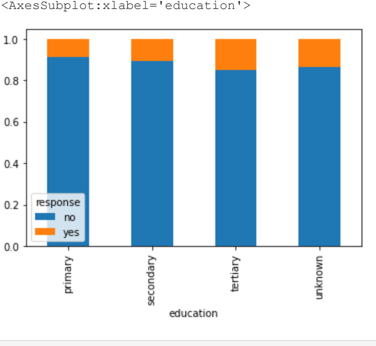
In [54]:
```python
2755/(2755+24459)
```

Out[54]:
```
0.10123465863158668
```

- OBSERVATION : marital status has no much significant difference on ressponse, single tend to respond positively more

In [55]:
```python
bdata.groupby('education')['response'].value_counts(normalize = True).unstack('respon
```

Out[55]:
```
<AxesSubplot:xlabel='education'>
```



In [56]:
```python
bdata.groupby('education')['response'].value_counts()
```

Out[56]:
```
education  response
primary    no           6260
           yes           591
secondary  no          20752
           yes          2450
tertiary   no          11305
           yes          1996
unknown    no           1605
           yes           252
Name: response, dtype: int64
```

In [57]:
```python
591/(591+6260)
```

Out[57]:
```
0.08626477886439936
```

In [58]:
```python
2450/(2450+20752)
```

Out[58]:
```
0.10559434531505904
```

In [59]:
```python
1996/(1996+11305)
```

Out[59]:
```
0.15006390496955116
```

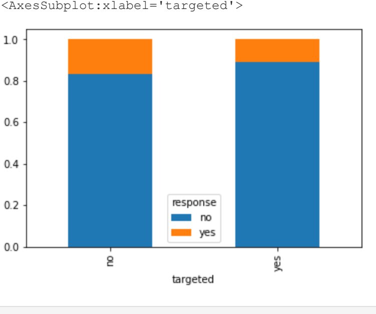
In [60]:
```python
252/(252+1605)
```

Out[60]:
```
0.13570274636510501
```

- OBSERVATION : tertary educated is more likely to respond positively

In [61]:
```python
bdata.groupby('targeted')['response'].value_counts(normalize = True).unstack('respons
```

Out[61]:
```
<AxesSubplot:xlabel='targeted'>
```



In [62]:
```python
bdata['targeted'].value_counts()
```

Out[62]:
```
yes    37091
no      8120
Name: targeted, dtype: int64
```

- OBSERVATION : Customer not tageted previously tend to respond more positively

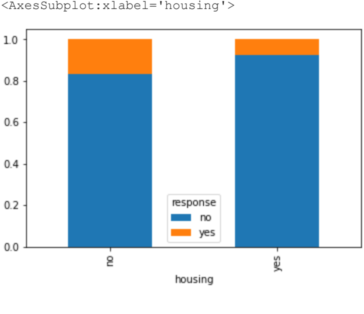
EFFECT of account balance on response variable

tertiary educated group has most median account balance and this group tends to more positive response than other

EFFECTS OF HOUSING LOAN ON TERM DEPOSIT

In [63]:
```python
bdata.groupby('housing')['response'].value_counts(normalize = True).unstack('response
```

Out[63]:
```
<AxesSubplot:xlabel='housing'>
```

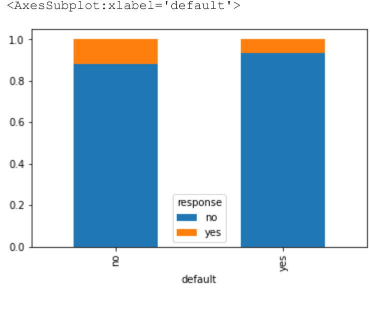


- OBSERVATION : if a customer has no housing loan tend to give more psitive response

EFFECT OF CREDIT DEFAULT ON RESPONSE RATE

In [64]:
```python
bdata.groupby('default')['response'].value_counts(normalize = True).unstack('response
```

Out[64]:
```
<AxesSubplot:xlabel='default'>
```

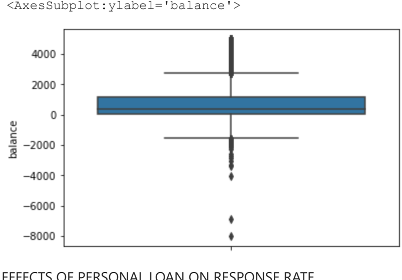


- Customer who has no any credit default tends to react positively more

In [66]:
```python
bbdata = bdata.loc[bdata['balance']  < 5000]
```

In [67]:
```python
sns.boxplot(y = 'balance', data = bbdata)
```

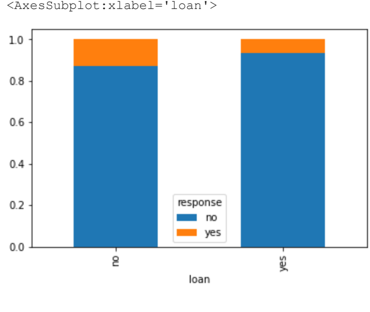
Out[67]:
```
<AxesSubplot:ylabel='balance'>
```



EFFECTS OF PERSONAL LOAN ON RESPONSE RATE

In [68]:
```python
bdata.groupby('loan')['response'].value_counts(normalize = True).unstack('response').p
```

Out[68]:
```
<AxesSubplot:xlabel='loan'>
```



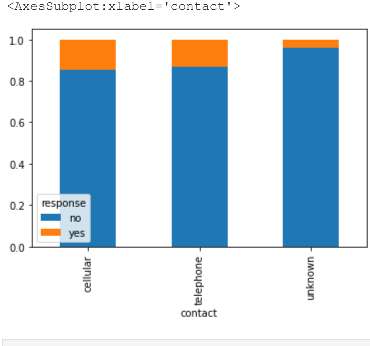
- customer who has no loan tends to respond positively more than double than who has loan

current campaign attribute

DOES COMMUNICATION TYPE HOLD ANY EFFECT ON RESPONSE VARIABLE

In [69]:
```python
bdata.groupby('contact')['response'].value_counts(normalize = True).unstack('response
```

Out[69]:
```
<AxesSubplot:xlabel='contact'>
```



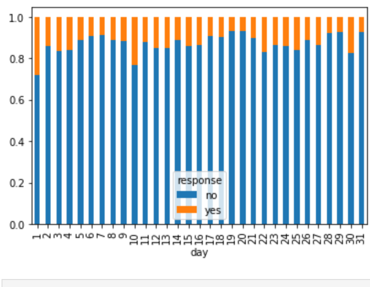
In [70]:
```python
bdata['contact'].value_counts()
```

Out[70]:
```
cellular     29285
unknown      13020
telephone     2906
Name: contact, dtype: int64
```

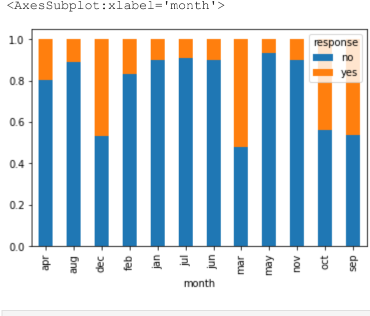
- OBSEVATION : communication type does not hold much effets on response variable

In [71]:
```python
bdata.groupby('day')['response'].value_counts(normalize = True).unstack('response').pl
```

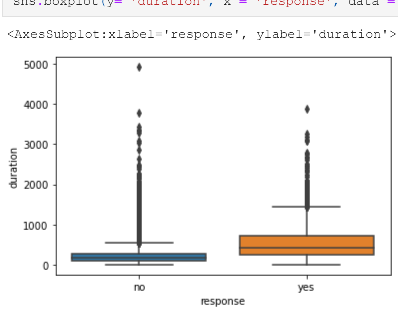
Out[71]:
```
<AxesSubplot:xlabel='day'>
```



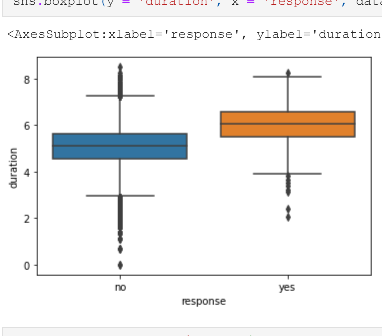
In [72]:
```
bdata.groupby('month')['response'].value_counts(normalize = True).unstack('response')
```
Out[72]: <AxesSubplot:xlabel='month'>



In [73]:
```
sns.boxplot(y= 'duration', x = 'response', data = bdata)
```
Out[73]: <AxesSubplot:xlabel='response', ylabel='duration'>



In [74]:
```
bdata = bdata.drop([6424,22937,36425], axis = 0)
```
In [77]:
```
bdata['duration'] = np.log(bdata['duration'])
```
In [81]:
```
sns.boxplot(y = 'duration', x = 'response', data = bdata)
```
Out[81]: <AxesSubplot:xlabel='response', ylabel='duration'>



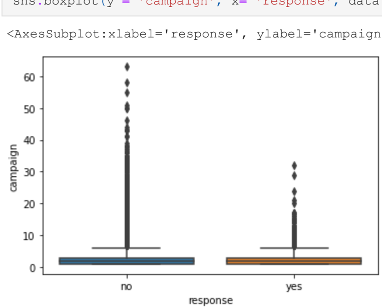
In [82]:
```
bdata.loc[bdata['duration']== 0]
```
Out[82]:

	age	job	salary	marital	education	targeted	default	balance	housing	loan	contact	day
31740	39	technician	60000	divorced	secondary	yes	no	17441	yes	yes	cellular	7
44758	28	admin.	50000	married	secondary	yes	no	1595	no	no	unknown	9

- duration has very high impact on rate of response

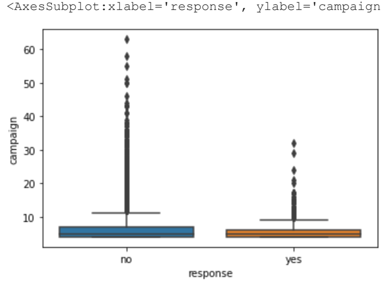
EFFECT OF NUMBER OF CONTACT ON RESPONSE RATE

In [86]:
```
sns.boxplot(y = 'campaign', x= 'response', data = bdata)
```
Out[86]: <AxesSubplot:xlabel='response', ylabel='campaign'>



In [87]:
```
bdata_5  = bdata.loc[bdata['campaign'] > 3]
```
In [89]:
```
35569/45211
```
Out[89]: 0.7867333171130919

In [90]:
```
sns.boxplot(y = 'campaign', x= 'response', data = bdata_5)
```
Out[90]: <AxesSubplot:xlabel='response', ylabel='campaign'>

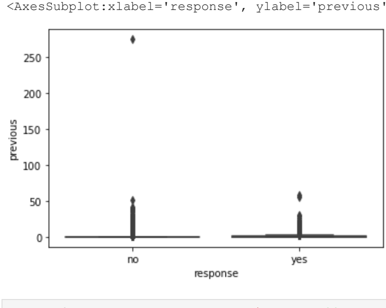


- number of campaign on response rate does not have much effects

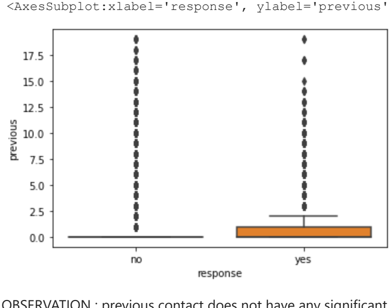
previous campaign attribute

EFFECTS OF NUMBER OF PREVIOUS CONTACT MADE TO THIS CLIENT BEFORE THIS COMPAIGN : Previous

In [92]:
```
sns.boxplot(y = 'previous', x= 'response', data = bdata)
```
Out[92]: <AxesSubplot:xlabel='response', ylabel='previous'>

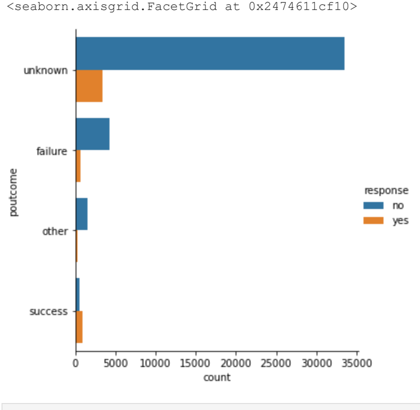


In [93]:
```
bdata_6 = bdata.loc[bdata['previous'] < 20 ]
```
In [94]:
```
sns.boxplot(y = 'previous', x= 'response', data = bdata_6)
```
Out[94]: <AxesSubplot:xlabel='response', ylabel='previous'>

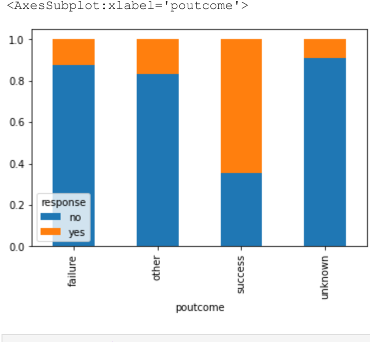


OBSERVATION : previous contact does not have any significant effects on response rate

In [97]:
```
sns.catplot(y = 'poutcome',hue = 'response', data = bdata, kind = 'count')
```
Out[97]: <seaborn.axisgrid.FacetGrid at 0x2474611cf10>



In [98]:
```
bdata.groupby('poutcome')['response'].value_counts(normalize = True).unstack('response
```
Out[98]: <AxesSubplot:xlabel='poutcome'>



In [99]:
```
(36959+1840)/45211
```
Out[99]: 0.8581761075844374

In [100…
```
bdata.groupby('poutcome')['response'].value_counts()
```
Out[100…
```
poutcome  response
failure   no           4283
          yes           618
other     no           1533
          yes           307
success   yes           978
          no            533
unknown   no          33570
          yes          3386
Name: response, dtype: int64
```
In [101…
```
978/(978+533)
```
Out[101… 0.6472534745201853

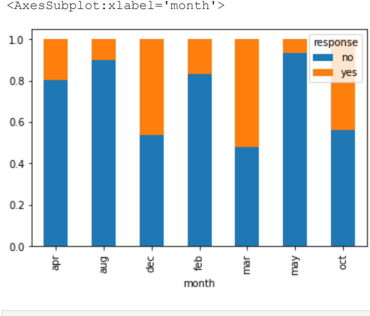
In [102…
```
bdata['response'].value_counts()
```
Out[102…
```
no     39919
yes     5289
Name: response, dtype: int64
```
In [103…
```
5289/(5289+39922)
```
Out[103… 0.11698480458295547

In [104…
```
bdata.loc[bdata['pdays'] < 1].shape
```
Out[104… (36951, 19)

In [107…
```
fdata = bdata.copy()
```
In [110…
```
fdata = fdata.drop(['pdays', 'previous','poutcome'], axis = 1)
```
In [112…
```
fdata = fdata.drop(['age'], axis = 1)
```
In [114…
```
fdata['job'].unique()
```
Out[114…
```
array(['management', 'technician', 'entrepreneur', 'blue-collar',
       'unknown', 'retired', 'admin.', 'services', 'self-employed',
       'unemployed', 'housemaid', 'student'], dtype=object)
```
In [115…
```
fdata['job'].replace({"admin.":"ASeUT","self-employed":"ASeUT","unknown":"ASeUT","tech
```
In [117…
```
fdata['salary'].replace({16000:18000,20000:18000}, inplace = True)
```
In [120…
```
fdata['marital'].replace({"married":"MD","divorced":"MD"}, inplace = True)
```
In [123…
```
fdata['education'].replace({"primary":"per_se","secondary":"per_se","unknown":"tertia
```
In [125…
```
fdata = fdata.drop(['balance'], axis = 1)
```
In [127…
```
fdata = fdata.drop(['contact'], axis = 1)
```
In [129…
```
fdata['day'].replace({13:12}, inplace = True)
```
In [131…
```
fdata['day'].replace({4:3,25:3}, inplace = True)
```
In [132…
```
fdata['day'].replace({7:6,17:6,18:6,28:6,29:6,31:6,21:6}, inplace = True)
```
In [134…
```
fdata['day'].replace({14:2,15:2,23:2,24:2,27:2,16:2}, inplace = True)
```
In [136…
```
fdata['day'].replace({8:5,9:5,11:5,19:5,20:5,26:5}, inplace = True)
```
In [138…
```
fdata['month'].replace({"sep":"dec"}, inplace = True)
```
In [139…
```
fdata['month'].replace({"jan":"aug","jul":"aug","jun":"aug","nov":"aug"}, inplace = Tr
```
In [142…
```
fdata = fdata.drop(['campaign'], axis = 1)
```

copy of data for modelling

In [147…
```
gdata = fdata.copy()
```
In [149…
```
fdata.job = fdata.job.astype('category')
```
In [151…
```
fdata.marital = fdata.marital.astype('category')
```
In [152…
```
fdata.education = fdata.education.astype('category')
```
In [153…
```
fdata.targeted = fdata.targeted.astype('category')
```
In [154…
```
fdata.default = fdata.default.astype('category')
```
In [155…
```
fdata.housing = fdata.housing.astype('category')
```
In [156…
```
fdata.loan = fdata.loan.astype('category')
```
In [157…
```
fdata.month = fdata.month.astype('category')
```
In [158…
```
fdata.day = fdata.day.astype('category')
```
In [194…
```
fdata.groupby('month')['response'].value_counts(normalize = True).unstack('response')
```
Out[194… <AxesSubplot:xlabel='month'>



In [161…
```
def target(row):
    if row['response'] ==  'yes':
        return 1
    else:
        return 0

fdata['RESPONSE'] = fdata.apply(target, axis = 1)
```

Three things to do respectively for improving the accuracy of model

1. Principal Component Analysis 2.Randomoversampling And Undersampling 3.Hyperparametretuning

train_test_split

In [164…
```
x = fdata.drop(['response','RESPONSE'],axis = 1)
y = fdata['RESPONSE']
```
In [167…
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.25, random_state
```

separation of categorical variables and numerical variables

In [170…
```
cat_att = list(x_train.select_dtypes(include = ['category']).columns)
```
In [172…
```
num_attr = list(x_train.select_dtypes(include = ['number']).columns)
num_attr
```
Out[172… ['salary', 'duration']

standardizing numerical data

In [173…
```
scaler = StandardScaler()
scaler.fit(x_train[num_attr])
x_train_std = pd.DataFrame(scaler.transform(x_train[num_attr]), columns = num_attr)
x_test_std = pd.DataFrame(scaler.transform(x_test[num_attr]),columns = num_attr)
```

one-hot encoding of categorical data

In [176…
```
ohe = OneHotEncoder()
ohe.fit(x_train[cat_att])
columns_ohe = list(ohe.get_feature_names(cat_att))
print(columns_ohe)

x_train_cat = ohe.transform(x_train[cat_att])
x_test_cat = ohe.transform(x_test[cat_att])

x_train_cat = pd.DataFrame(x_train_cat.todense(), columns = columns_ohe)
x_test_cat = pd.DataFrame(x_test_cat.todense(), columns = columns_ohe)
```
```
['job_ASeUT', 'job_SHE', 'job_blue-collar', 'job_management', 'job_retired', 'job_stud
ent', 'job_unemployed', 'marital_MD', 'marital_single', 'education_per_se', 'education
_tertiary', 'targeted_no', 'targeted_yes', 'default_no', 'default_yes', 'housing_no',
'housing_yes', 'loan_no', 'loan_yes', 'day_1', 'day_2', 'day_3', 'day_5', 'day_6', 'da
y_10', 'day_12', 'day_22', 'day_30', 'month_apr', 'month_aug', 'month_dec', 'month_fe
b', 'month_mar', 'month_may', 'month_oct']
```
In [177…
```
x_train_prep = pd.concat([x_train_std,x_train_cat], axis = 1)
x_test_prep = pd.concat([x_test_std, x_test_cat], axis =1)
```
In [180…
```
from imblearn.over_sampling import RandomOverSampler
```
In [181…
```
ros = RandomOverSampler()
x_ros , y_ros = ros.fit_resample(x_train_prep, y_train)
```
In [184…
```
LR = LogisticRegression()
LR.fit(x_ros, y_ros)
```
Out[184… LogisticRegression()

In [187…
```
confusion_matrix(y_ros, y_pred_train_LR)
```
Out[187…
```
array([[23775,  6164],
       [ 5297, 24642]], dtype=int64)
```
In [ ]:
```
plot_confusion_matrix(LR,x_ros,y_ros)
```
In [188…
```
y_pred_train_LR = LR.predict(x_ros)
print(classification_report(y_ros,y_pred_train_LR))
```
```
              precision    recall  f1-score   support

           0       0.82      0.79      0.81     29939
           1       0.80      0.82      0.81     29939

    accuracy                           0.81     59878
   macro avg       0.81      0.81      0.81     59878
weighted avg       0.81      0.81      0.81     59878
```

test data

In [189…
```
y_pred_test_LR = LR.predict(x_test_prep)
print(classification_report(y_test, y_pred_test_LR))
```
```
              precision    recall  f1-score   support

           0       0.97      0.79      0.87      9980
           1       0.34      0.81      0.48      1322

    accuracy                           0.79     11302
   macro avg       0.65      0.80      0.68     11302
weighted avg       0.90      0.79      0.83     11302
```

RESULT : Since recall is more important matrics in this business problem, model getting 81% recall rate