Introduction

Data Set Information:

The data is related with direct marketing campaigns of a European banking institution. The marketing campaigns were based on phone calls and were for marketing their product - bank term deposit. The information of client subscribing to the product is indicated in the column y which is either yes or no. Information about a predictive model currently being used by the bank is also included. The dataset contains 41188 records with 20 inputs, ordered by date (from May 2008 to November 2010).

Problem Statement:

The task is to study the data, find data patterns and clean the data for further modeling to help this banking institution determine, in advance, clients who will be receptive to such marketing campaigns. Evaluate performance of the deployed model, state the evaluation metric used and suggest whether the bank ought to replace the current model.

Attribute Information:

Input variables:
1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical:
'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown') # related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model. # other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success') # social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)
21- ModelPerformance – Results from a current model used to predict whether a client will subscribe (‘yes’) to a term deposit (probability of subscribing to term deposit)
22 - Output Variable (desired target): y - has the client subscribed a term deposit? (binary: 'yes','no')

In [1]:
```
#Importing Data Analysis Libraries

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from plotly import tools
import plotly.figure_factory as ff
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

import warnings
warnings.filterwarnings('ignore')
import os
```

In [2]:
```
#import datasheet

df = pd.read_csv('DSA Data Set.csv')
```

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

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon
1	57	services	married	high.school	unknown	no	no	telephone	may	mon
2	37	services	married	high.school	no	yes	no	telephone	may	mon
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon
4	56	services	married	high.school	no	no	yes	telephone	may	mon

5 rows × 22 columns

Data Cleaning

In [3]:
```
#Change column name: 'y' to 'response'
df.rename(index=str, columns={'y': 'response'}, inplace = True)
```

In [4]:
```
#Checking for null or missing values
df.info()
#All the columns are non-null, no missing values
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 41188 entries, 0 to 41187
Data columns (total 22 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  ModelPrediction  41188 non-null  float64
 21  response         41188 non-null  object 
dtypes: float64(6), int64(5), object(11)
memory usage: 7.2+ MB
```

All the columns are non-null, no missing values

In [8]:
```
#checking for duplicated rows
duplicate = df[df.duplicated()]
duplicate
```

Out[8]:

	age	job	marital	education	default	housing	loan	contact	month	d
1266	39	blue-collar	married	basic.6y	no	no	no	telephone	may	
12261	36	retired	married	unknown	no	no	no	telephone	jul	
14234	27	technician	single	professional.course	no	no	no	cellular	jul	
16956	47	technician	divorced	high.school	no	yes	no	cellular	jul	
18465	32	technician	single	professional.course	no	yes	no	cellular	jul	
20216	55	services	married	high.school	unknown	no	no	cellular	aug	
20534	41	technician	married	professional.course	no	yes	no	cellular	aug	
25217	39	admin.	married	university.degree	no	no	no	cellular	nov	
28477	24	services	single	high.school	no	yes	no	cellular	apr	
32516	35	admin.	married	university.degree	no	yes	no	cellular	may	
36951	45	admin.	married	university.degree	no	no	no	cellular	jul	
38281	71	retired	single	university.degree	no	no	no	telephone	oct	

12 rows × 22 columns

No duplicate rows

Exploratory Data Analysis

In [9]:
```
#Statistical analysis of numerical columns of the dataset
df.describe()
```

Out[9]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	4118
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	9
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	9
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	9
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	9
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	9
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	9

In [10]:
```
#Understanding the unique values per column
print('Job:\n', df['job'].unique())
print('Marital Status:\n', df['marital'].unique())
print('Education:\n', df['education'].unique())
print('Default:\n', df['default'].unique())
print('Housing:\n', df['housing'].unique())
print('Loan:\n', df['loan'].unique())
print('Contact:\n', df['contact'].unique())
print('Month:\n', df['month'].unique())
print('Day of Week:\n', df['day_of_week'].unique())
print('Campaign:\n', df['campaign'].unique())
print('Pdays:\n', df['pdays'].unique())
print('Previous:\n', df['previous'].unique())
print('Subscribed:\n', df['response'].unique())
```

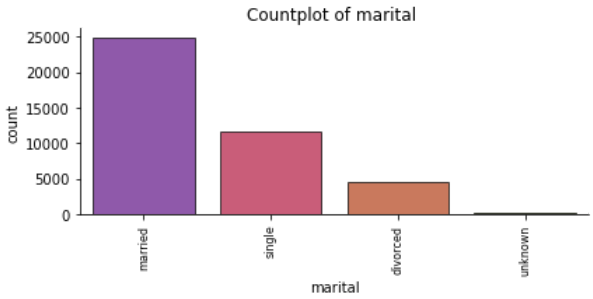
```
Job:
 ['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']
Marital Status:
 ['married' 'single' 'divorced' 'unknown']
Education:
 ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course'
 'unknown' 'university.degree' 'illiterate']
Default:
 ['no' 'unknown' 'yes']
Housing:
 ['no' 'yes' 'unknown']
Loan:
 ['no' 'yes' 'unknown']
Contact:
 ['telephone' 'cellular']
Month:
 ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'apr' 'sep']
Day of Week:
 ['mon' 'tue' 'wed' 'thu' 'fri']
Campaign:
 [ 1  2  3  4  5  6  7  8  9 10 11 12 13 19 18 23 14 22 25 16 17 15 20 56
 39 35 42 28 26 27 32 21 24 29 31 30 41 37 40 33 34 43]
Pdays:
 [999   6   4   3   5   1   0  10   7   8   9  11   2  12  13  14  15  16
  21  17  18  22  25  26  19  27  20]
Previous:
 [0 1 2 3 4 5 6 7]
Subscribed:
 ['no' 'yes']
```

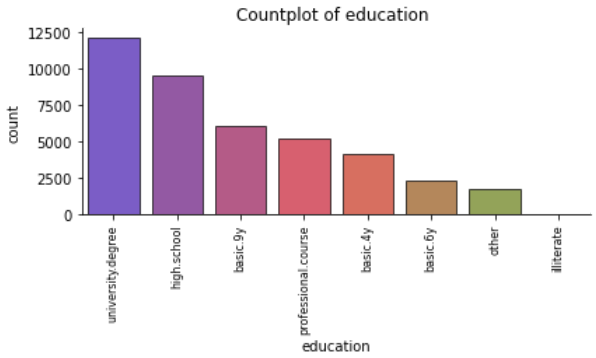
In [11]:
```
#Cleaning - Replacing the unknown values in job and education variables as oth
er
df[['job','education']] = df[['job', 'education']].replace(['unknown'],'other'
)
```

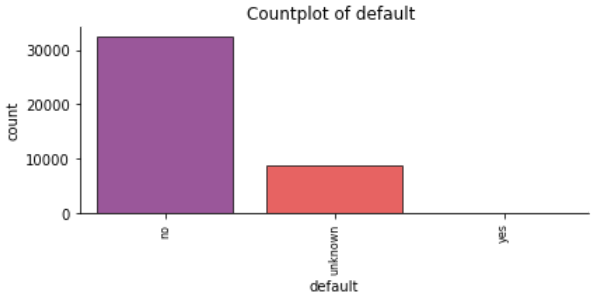
In [12]:
```
print('Job:\n', df['job'].unique())
print('Education:\n', df['education'].unique())
```

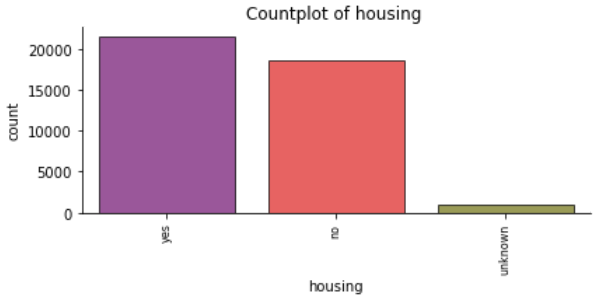
```
Job:
 ['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' 'other' 'entrepreneur'
 'student']
Education:
 ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course'
 'other' 'university.degree' 'illiterate']
```

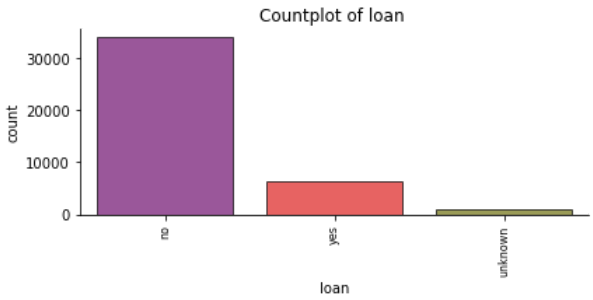
In [13]:
```
#Distribution of Categorical Variables
plt.figure(figsize=[12,14])
cat_variables=["marital", "education", "default", "housing", "loan", "poutcom
e","contact", "month"]
n=1
for i in cat_variables:
    order1 = df[i].value_counts(ascending=False).index
    plt.subplot(4,2,n)
    sns.countplot(x=i, edgecolor="black", alpha=0.7, data=df, order = order1,
palette= 'brg')
    plt.xticks(fontsize=8,rotation=90)
    sns.despine()
    plt.title("Countplot of {} ".format(i))
    n=n+1
plt.tight_layout()
plt.show()


plt.figure(figsize=[14,4])
order1 = df['job'].value_counts(ascending=False).index
sns.countplot(x='job',edgecolor="black", alpha=0.7, data=df, palette= 'brg', o
rder = order1)
plt.xticks(fontsize=10,rotation=90)
sns.despine()
plt.title("Countplot of job")
plt.show()
```

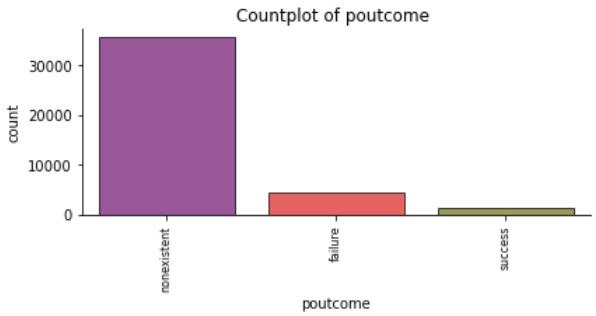

Countplot of marital
count
marital
married
single
divorced
unknown

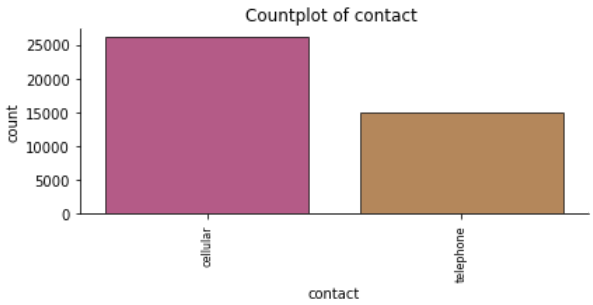

Countplot of education
count
education
university.degree
high.school
basic.9y
professional.course
basic.4y
basic.6y
other
illiterate

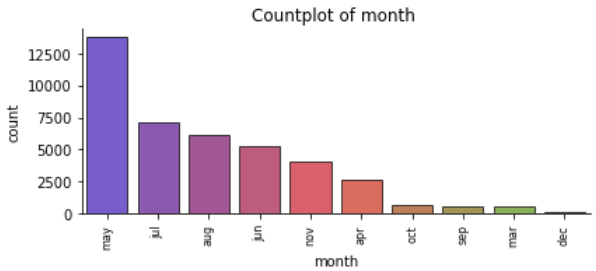

Countplot of default
count
default
no
unknown
yes

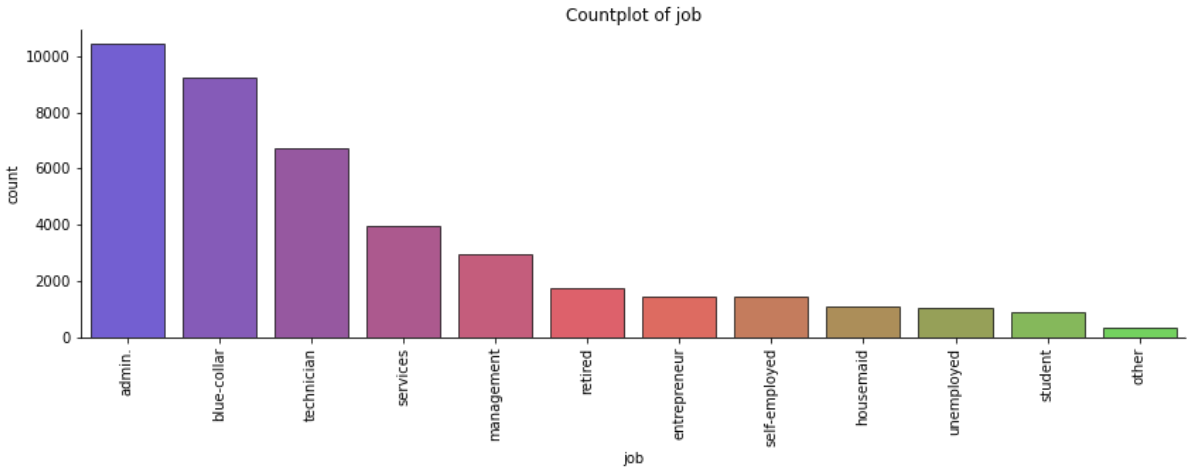

Countplot of housing
count
housing
yes
no
unknown


Countplot of loan
count
loan
no
yes
unknown


Countplot of poutcome
count
poutcome
nonexistent
failure
success

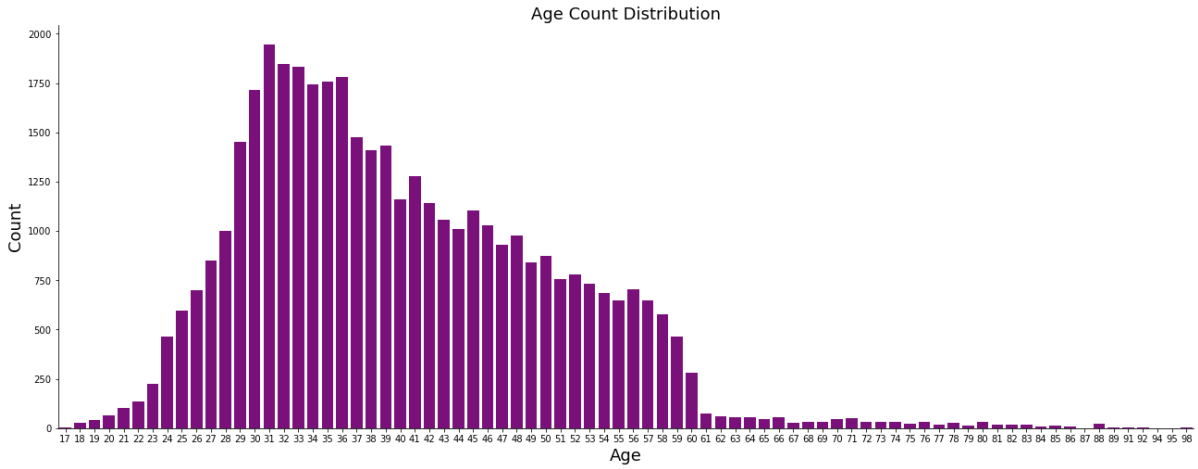

Countplot of contact
count
contact
cellular
telephone


Countplot of month
count
month
may
jul
aug
jun
nov
apr
oct
sep
mar
dec

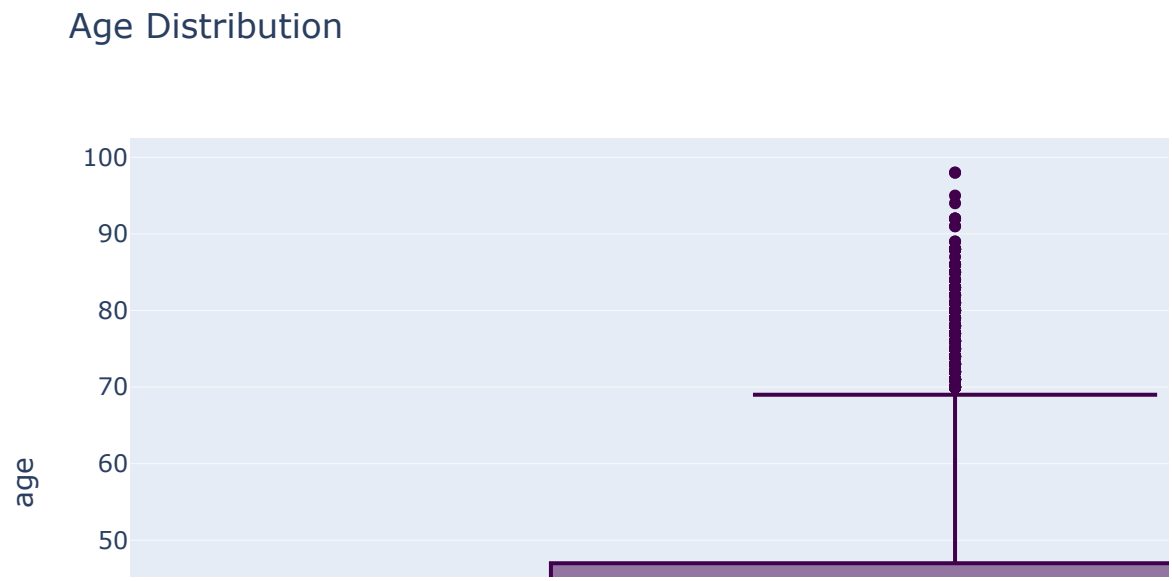

Countplot of job
count
job
admin.
blue-collar
technician
services
management
retired
entrepreneur
self-employed
housemaid
unemployed
student
other

For Numerical Columns

In [14]:
```
#For understanding age distribution
fig, ax = plt.subplots()
fig.set_size_inches(22, 8)
sns.countplot(x = 'age', data = df, color = "darkmagenta")
ax.set_xlabel('Age', fontsize=18)
ax.set_ylabel('Count', fontsize=18)
ax.set_title('Age Count Distribution', fontsize=18)
sns.despine()
```



In [7]:
```
import plotly.express as px
fig = px.box(df, y="age", title = "Age Distribution",
             color_discrete_sequence= px.colors.diverging.PRGn)
fig.show()
```



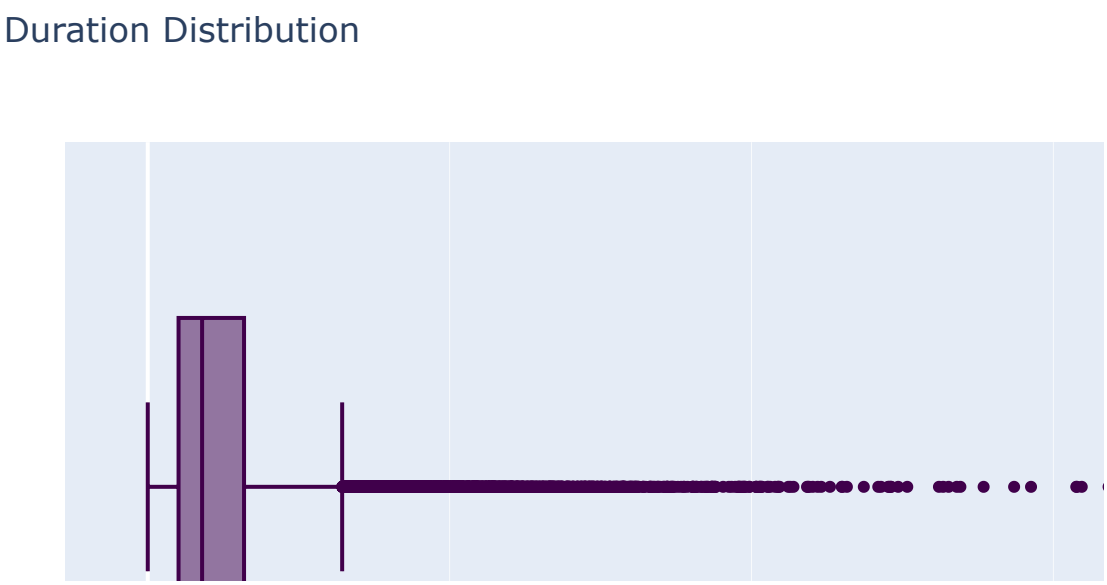
Analysis of Age Distribution

The distribution of age: In its telemarketing campaigns, clients called by the bank have an extensive age range, from 17 to 98 years old. However, a majority of customers called is in the age of 30s and 40s (32 to 47 years old fall within the 25th to 75th percentiles). The distribution of customer age is fairly normal with a small standard deviation.

In [78]:
```
df['age'].describe()
```

Out[78]:
```
count    41188.00000
mean        40.02406
std         10.42125
min         17.00000
25%         32.00000
50%         38.00000
75%         47.00000
max         98.00000
Name: age, dtype: float64
```

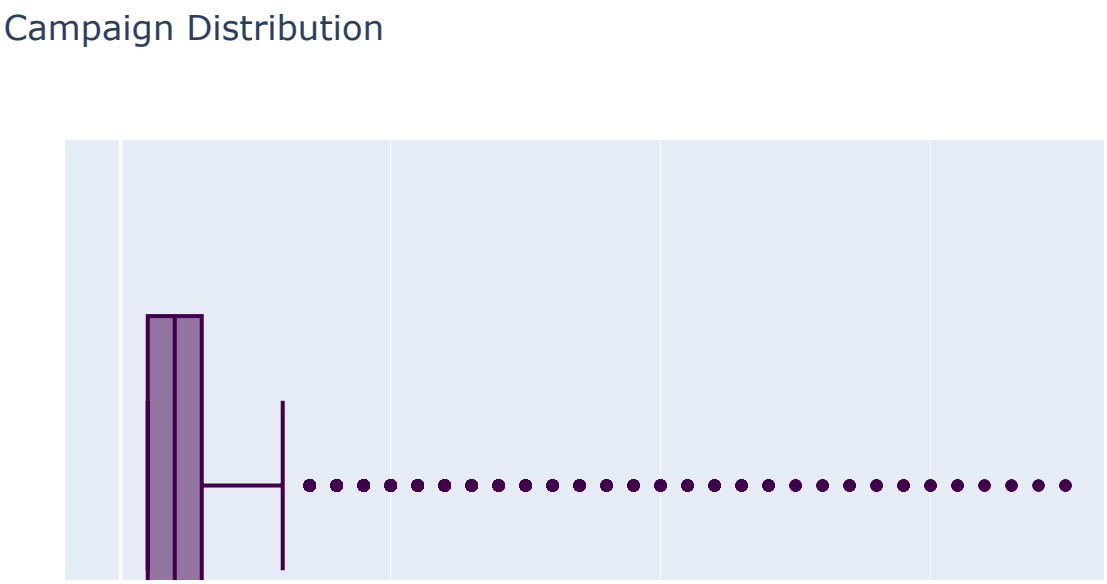
In [27]:
```
fig = px.box(df, x="duration", title = "Duration Distribution", color_discrete
_sequence= px.colors.diverging.PRGn)
fig.show()
```



Analysis for Duration

As observed from the box plot, the duration of contact has a median of 3 minutes, with an interquartile range of 1.73 minutes to 5.3 minutes. The left-skewed boxplot indicates that most calls are relatively short. Also, there is a large number of outliers ranging from 10 minutes to 40 minutes.

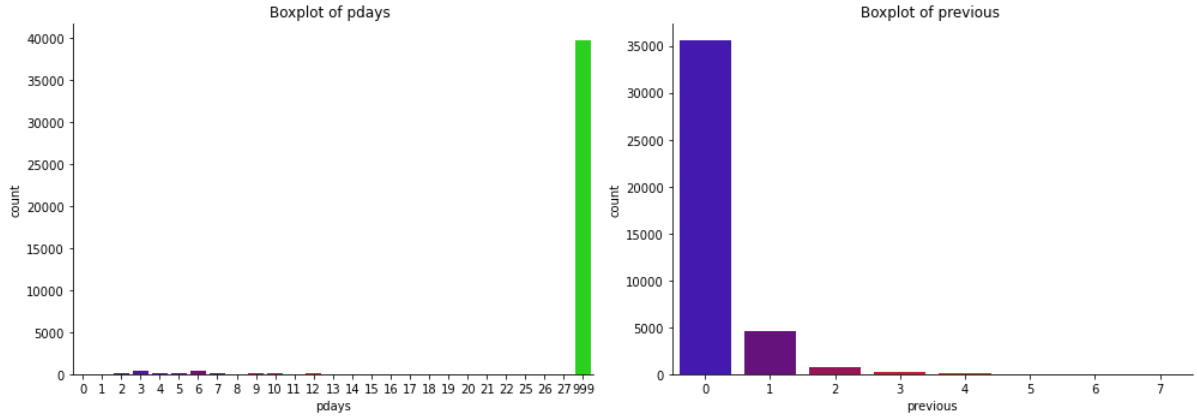
In [28]:
```
fig = px.box(df, x="campaign", title = "Campaign Distribution", color_discrete
_sequence= px.colors.diverging.PRGn)
fig.show()
```



Analysis for Campaign Distribution

About half of the clients have been contacted by the bank for the second time, while 25% was first introduced to the term deposit. Most clients have been reached by the bank for one to three times, which is reasonable. However, some clients have been contacted by as high as 56 times, which is not normal. These clients may have some special needs that require frequent contact.

In [32]:
```
#Distribution of previous marketing attributes
plt.figure(figsize=[14,5])
socio_eco_variables=["pdays", "previous"]
n=1
for i in socio_eco_variables:
    plt.subplot(1,2,n)
    sns.countplot(x=i, data=df, palette = 'brg')
    sns.despine()
    plt.title("Boxplot of {} ".format(i))
    n=n+1
plt.tight_layout()
plt.show()
#tableau refer
```

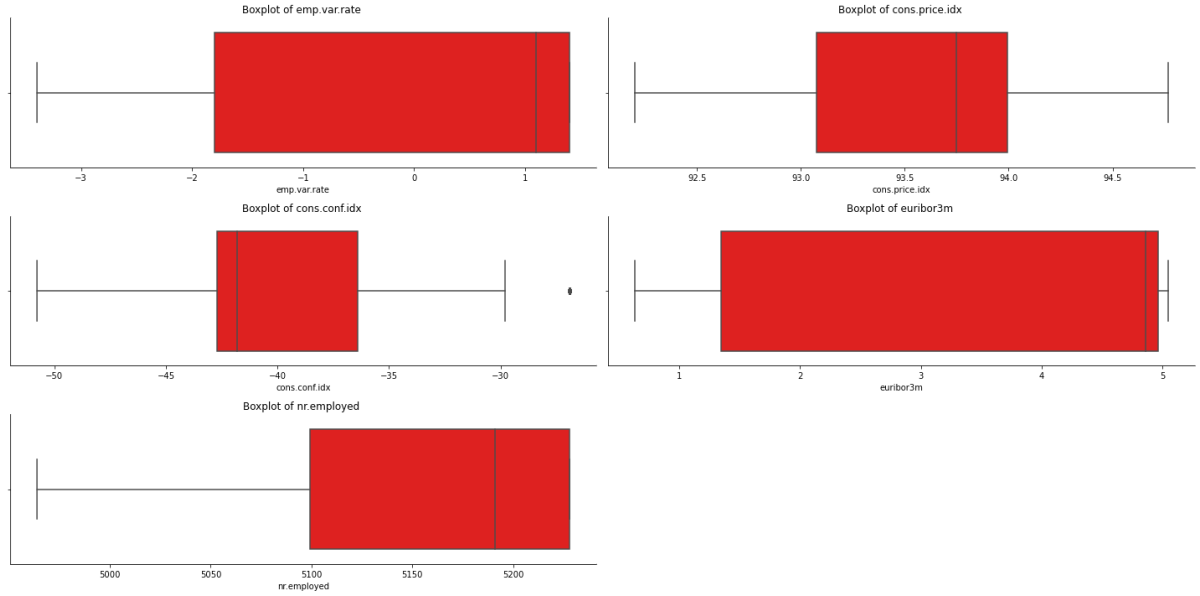


Observation:

Pdays - Number of days that passed by after the client was last contacted from a previous campaign

Previous - Number of contacts performed before this campaign and for this client have no effect on response.

In [102]:
```
#Distribution of social and economic context attributes
plt.figure(figsize=[20,10])
socio_eco_variables=["emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed"]
n=1
for i in socio_eco_variables:
    plt.subplot(3,2,n)
    sns.boxplot(x=i, data=df, palette = 'brg')
    sns.despine()
    plt.title("Boxplot of {} ".format(i))
    n=n+1
plt.tight_layout()
plt.show()
```



In [59]:
```
df['cons.conf.idx'].describe()
```

Out[59]:
```
count    41188.000000
mean       -40.502600
std          4.628198
min        -50.800000
25%        -42.700000
50%        -41.800000
75%        -36.400000
max        -26.900000
Name: cons.conf.idx, dtype: float64
```

Analysis

emp.var.rate: employment variation rate - quarterly indicator

The variation of how many people are being hired or fired due to the shifts in the conditions of the economy. Economists consider a natural unemployment rate of around 4%-6% to be an indicator of a healthy economy.

cons.price.idx: consumer price index - monthly indicator

Determines cost of living changes, inflation (high cpi), purchasing power,

cons.conf.idx: consumer confidence index - monthly indicator

Consumer confidence, an economic indicator that measures the degree of optimism that consumers have regarding the overall state of a country's economy and their own financial situations. If the most recent index is above 100, then consumers are more confident than they were in 1985. If it's below 100, they are less confident than during that time.

euribor 3 month rate - daily indicator

The 3 month Euribor interest rate is the interest rate at which a panel of banks lend money to one another with a maturity of 3 months

Data Visualization

Visualizing each attribute with respect to the target column response to evaluate the performance of telemarketing campaign

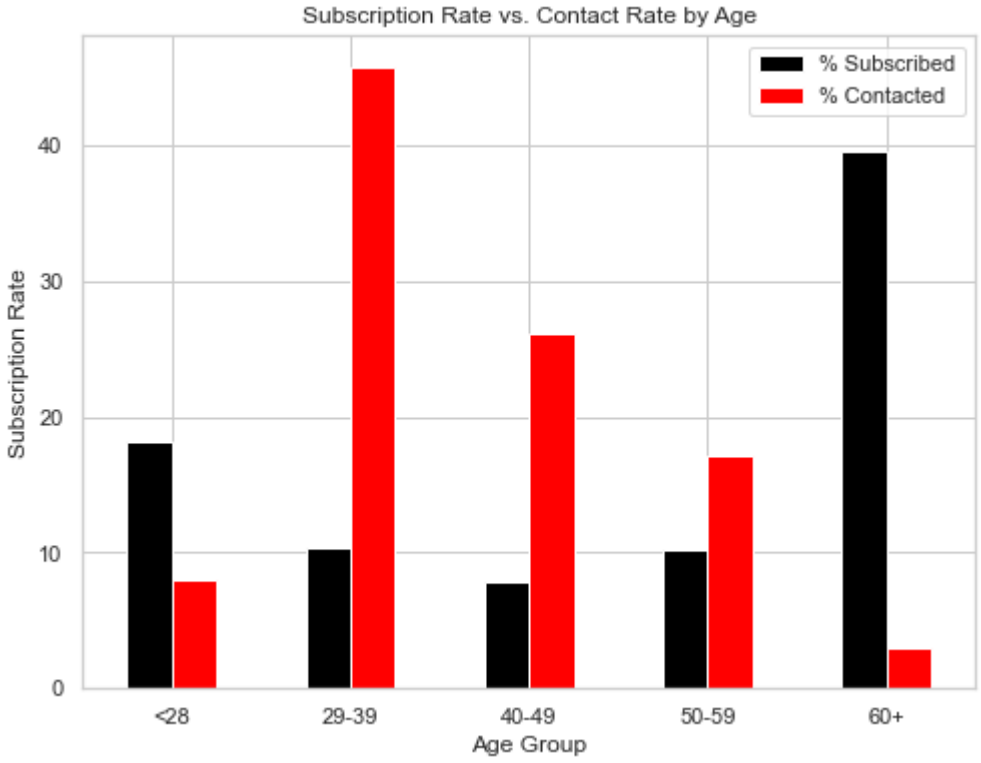
Age with respect to Response

In [116]:
```
#Age
age_range = [df]
for i in age_range:
    i.loc[i['age']<28, 'age_group'] = 20
    i.loc[(i['age']>=29) & (i['age']<=39), 'age_group'] = 30
    i.loc[(i['age']>=40) & (i['age']<=49), 'age_group'] = 40
    i.loc[(i['age']>=50) & (i['age']<=59), 'age_group'] = 50
    i.loc[i['age']>=60, 'age_group'] = 60
```

In [117]:
```
cnt_age_resp = pd.crosstab(df['response'],df['age_group']).apply(lambda x: x/x
.sum() * 100)
cnt_age_resp = cnt_age_resp.transpose()
```

In [118]:
```
age_df = pd.DataFrame(df['age_group'].value_counts())
age_df['% Contacted'] = age_df['age_group']*100/age_df['age_group'].sum()
age_df['% Subscribed'] = cnt_age_resp['yes']
age_df.drop('age_group',axis = 1,inplace = True)

age_df['age'] = [30,40,50,20,60]
age_df = age_df.sort_values('age',ascending = True)
```

In [121]:
```
g_age = age_df[['% Subscribed','% Contacted']].plot(kind = 'bar',
                                                    figsize=(8,6), color = ('black',
'red'))
plt.xlabel('Age Group')
plt.ylabel('Subscription Rate')
plt.xticks(np.arange(5), ('<28', '29-39', '40-49', '50-59', '60+'),rotation =
'horizontal')
plt.title('Subscription Rate vs. Contact Rate by Age')
plt.show()
```



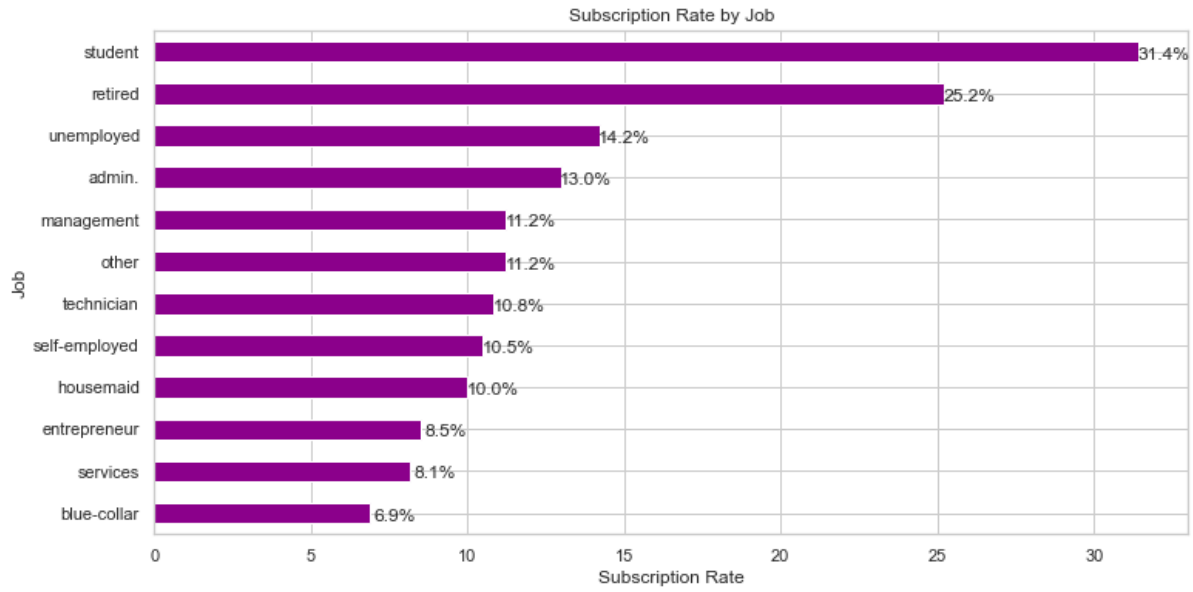
Insights: Target the youngest and the oldest instead of the middle-aged

The graph indicates that clients with a age of 60+ have the highest subscription rate at 39%. About 18% of the subscriptions came from the clients aged between 18 to 29. More than 50% of the subscriptions are contributed by the youngest and the eldest clients.

It is not surprising to see such a pattern because the main investment objective of older people is saving for retirement while the middle-aged group tend to be more aggressive with a main objective of generating high investment income. Term deposits, as the least risky investment tool, are more preferable to the eldest. The youngest may not have enough money or professional knowledge to engage in sophisticated investments, such as stocks and mutual funds. Term deposits provide liquidity and generate interest incomes that are higher than the regular saving account, so term deposits are ideal investments for students. However, red vertical bars show that the bank focused its marketing efforts on the middle-aged group, which returned lower subscription rates than the younger and older groups. Thus, to make the marketing campaign more effective, the bank should target younger and older clients in the future.

In [124]:
```
# Job
cnt_job_resp = pd.crosstab(df['response'],df['job']).apply(lambda x: x/x.sum()
* 100)
cnt_job_resp = cnt_job_resp.transpose()
```

In [126]:
```
g_job = cnt_job_resp['yes'].sort_values(ascending = True).plot(kind = 'barh',
figsize = (12,6), color = 'darkmagenta')

plt.title('Subscription Rate by Job')
plt.xlabel('Subscription Rate')
plt.ylabel('Job')

# Label each bar
for rec, label in zip(g_job.patches,
                      cnt_job_resp['yes'].sort_values(ascending = True).round(
1).astype(str)):
    g_job.text(rec.get_width()+0.8,
                   rec.get_y()+ rec.get_height()-0.5,
                   label+'%',
                   ha = 'center',
                   va='bottom')
```



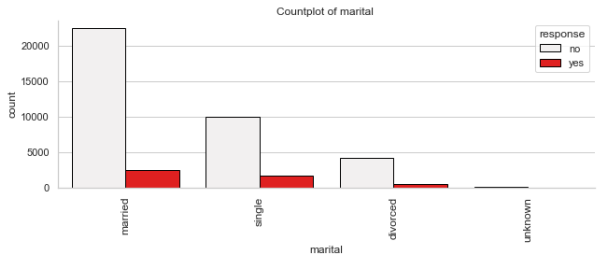
Observation

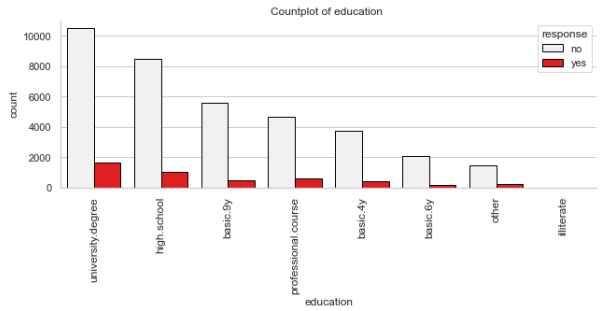
Job – Response graph agrees with the Age-Response graph showing Students and Retired clients response positive, followed by unemployed and white collar jobs

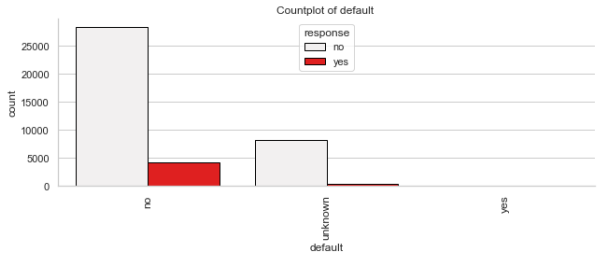
Insights

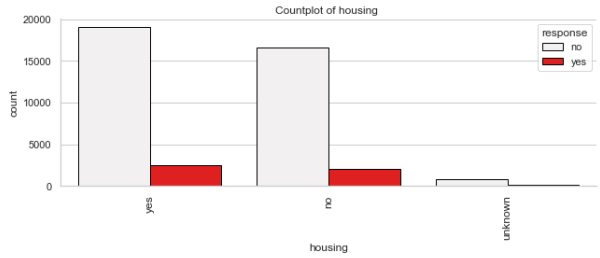
Target students, retired, unemployed clients more than clients who are admin or blue collared.

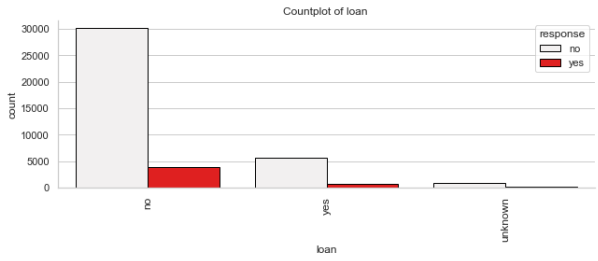
In [144]:
```
#Distribution of Categorical Variables
plt.figure(figsize=[18,18])
cat_variables=["marital", "education", "default", "housing", "loan", "poutcom
e","contact", "month"]
n=1
for i in cat_variables:
    order1 = df[i].value_counts(ascending=False).index
    plt.subplot(4,2,n)
    sns.countplot(x=i, edgecolor="black", alpha=1, data=df, order = order1, hu
e = 'response', color = 'red')
    plt.xticks(fontsize = 12 ,rotation=90)
    sns.despine()
    plt.title("Countplot of {} ".format(i))
    n=n+1
plt.tight_layout()
plt.show()


plt.figure(figsize=[18,4])
order1 = df['job'].value_counts(ascending=False).index
sns.countplot(x='job',edgecolor="black", alpha=0.7, data=df, hue = 'response',
order = order1, color = 'red')
plt.xticks(fontsize=10,rotation=90)
sns.despine()
plt.title("Countplot of job")
plt.show()
```

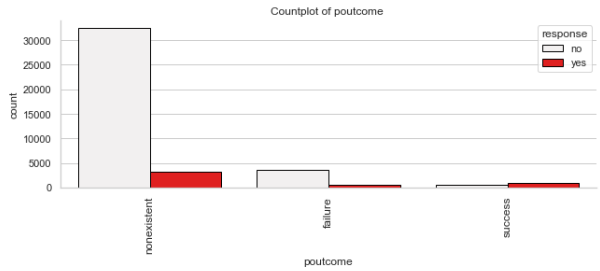

Countplot of marital
response
no
yes
count
marital
married
single
divorced
unknown

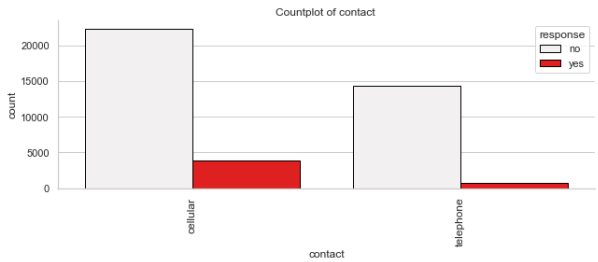

Countplot of education
response
no
yes
count
education
university.degree
high.school
basic.9y
professional.course
basic.4y
basic.6y
other
illiterate

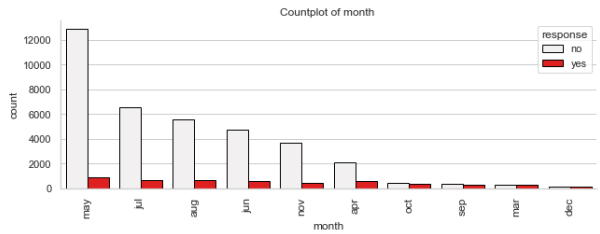

Countplot of default
response
no
yes
count
default
no
unknown
yes

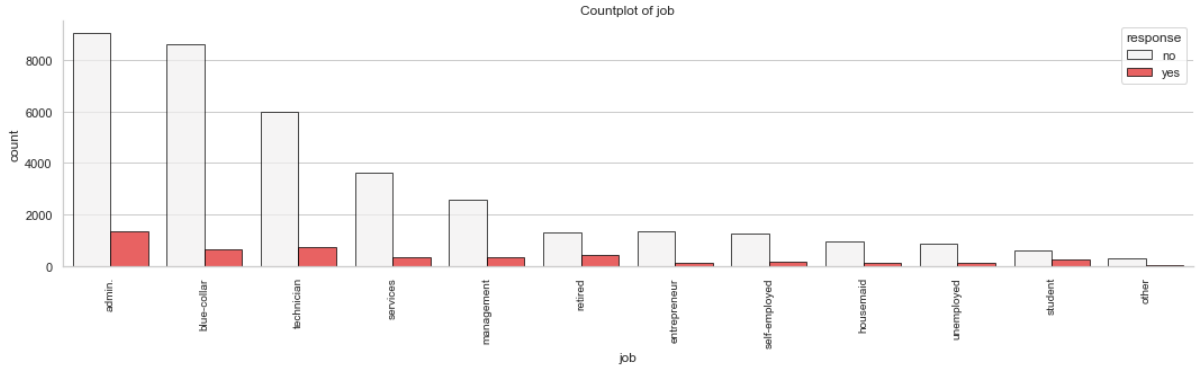

Countplot of housing
response
no
yes
count
housing
yes
no
unknown


Countplot of loan
response
no
yes
count
loan
no
yes
unknown


Countplot of poutcome
response
no
yes
count
poutcome
nonexistent
failure
success


Countplot of contact
response
no
yes
count
contact
cellular
telephone


Countplot of month
response
no
yes
count
month
may
jul
aug
jun
nov
apr
oct
sep
mar
dec


Countplot of job
response
no
yes
count
job
admin.
blue-collar
technician
services
management
retired
entrepreneur
self-employed
housemaid
unemployed
student
other

Observations:

In the education-response graph, university degree, professional course and unknown have higher conversion.

In marital – response graph, single clients followed by married have high conversion rate. Divorced clients show lowest conversion.

The default and housing graph does not show a strong relationship with response.

The personal loan graph indicates that clients without existing personal loans were contacted more than the ones with. But the percentage of positive response is equal in both cases.

Poutcome: Outcome of the previous marketing campaign shows clients who enrolled/interested in previous campaign had a positive response.

Insights:

Clients with personal loans must be contacted and equal number of clients with personal loans must be contacted to compare and support the observation that clients with PL have a higher chance of investing in FD.

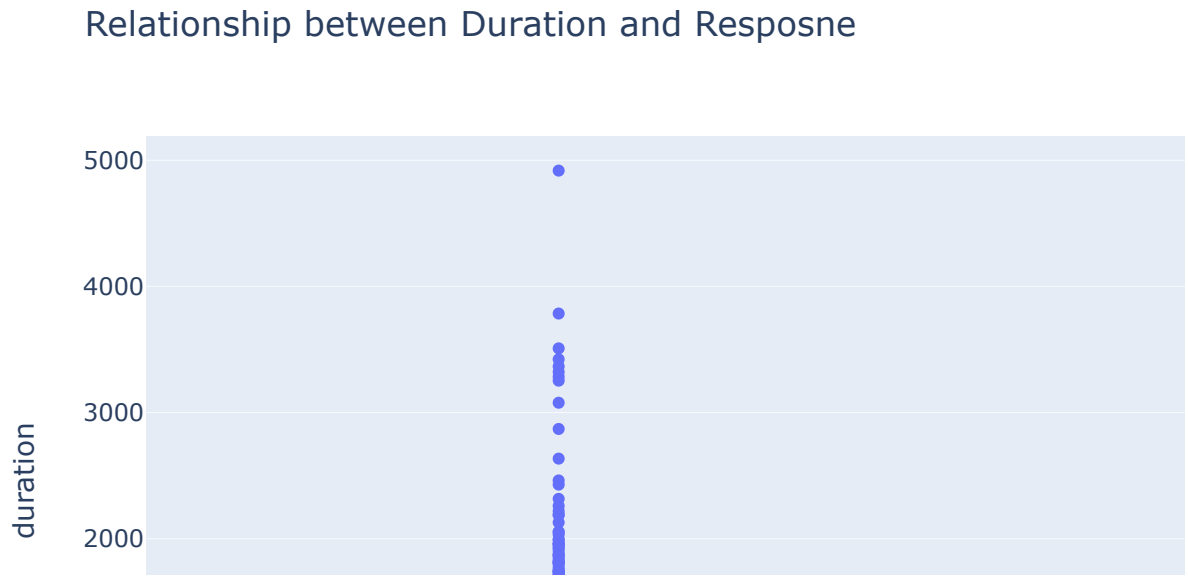
Clients who have shown interest in previous products should be contacted more.

Response of clients contacted on cellphone is more positive than telephone.

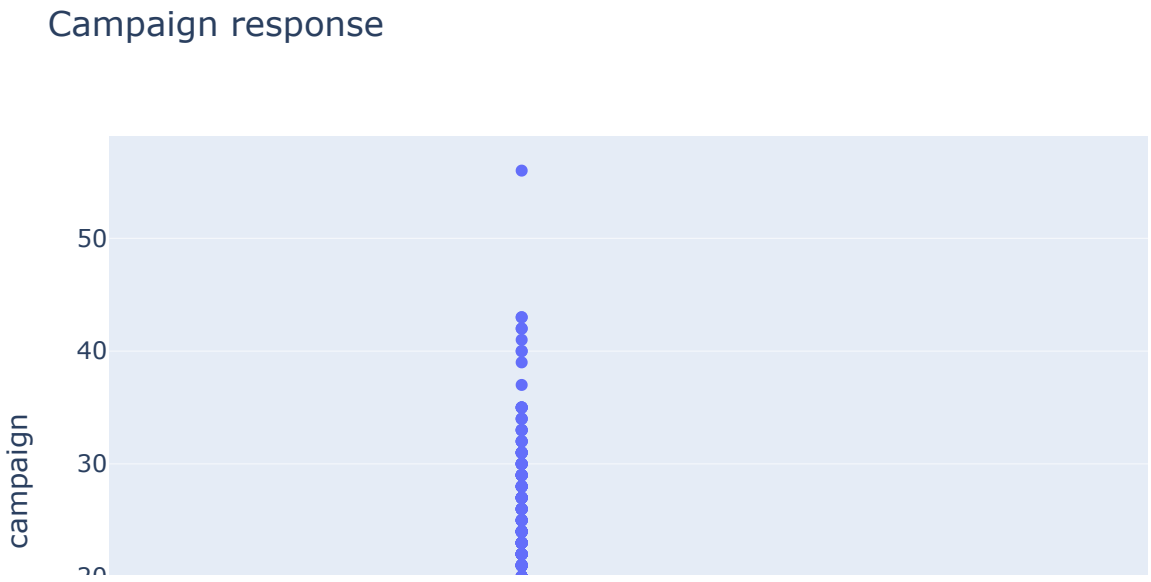
Target clients with higher level education, need to find what unknown indicate.

Target clients who are single, with second priority married client.

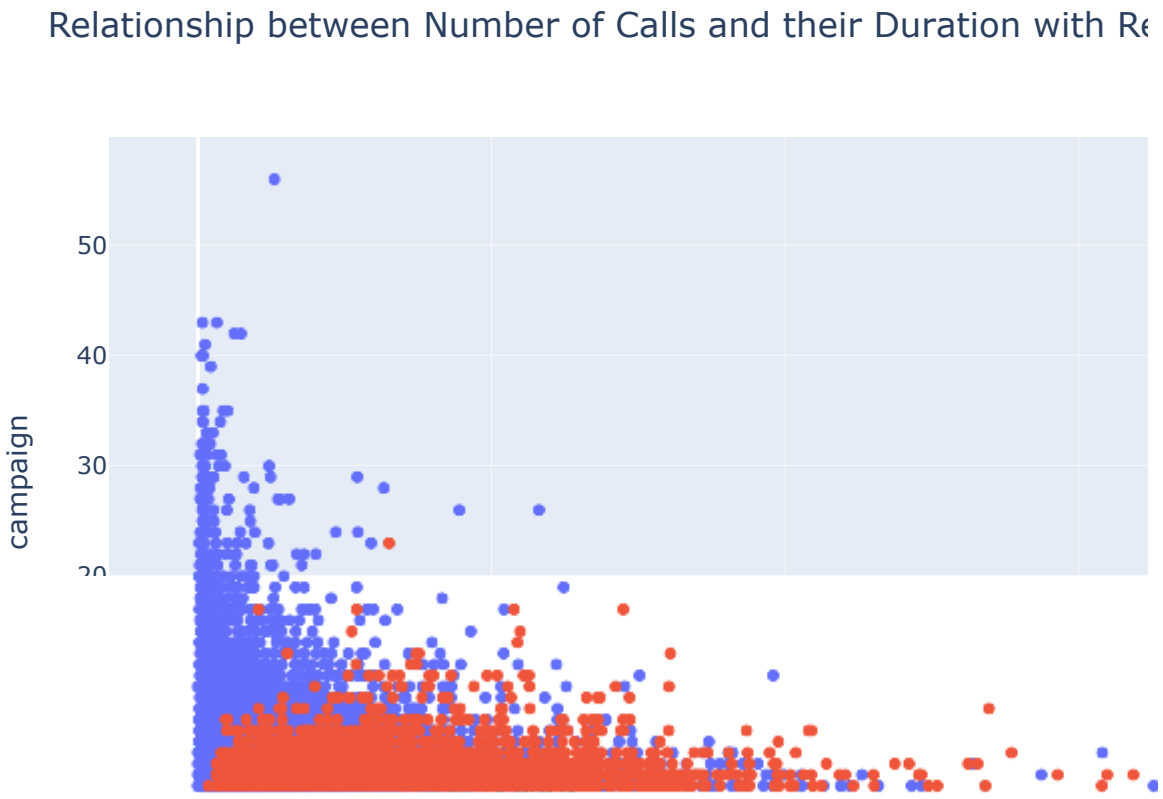
In [9]:
```
fig = px.box(df, y= 'duration', x="response", title = "Relationship between Du
ration and Resposne")
fig.show()
```



In [161]:
```
fig = px.box(df, y= 'campaign', x="response", title = "Campaign response")
fig.show()
```



In [14]:
```
#duration and response
fig = px.scatter(df, x="duration", y="campaign", color = 'response',
                 title= 'Relationship between Number of Calls and their Durati
on with Response',
                 labels= {'x':'Duration in seconds', 'y':'Number of Calls'})
fig.show()
```



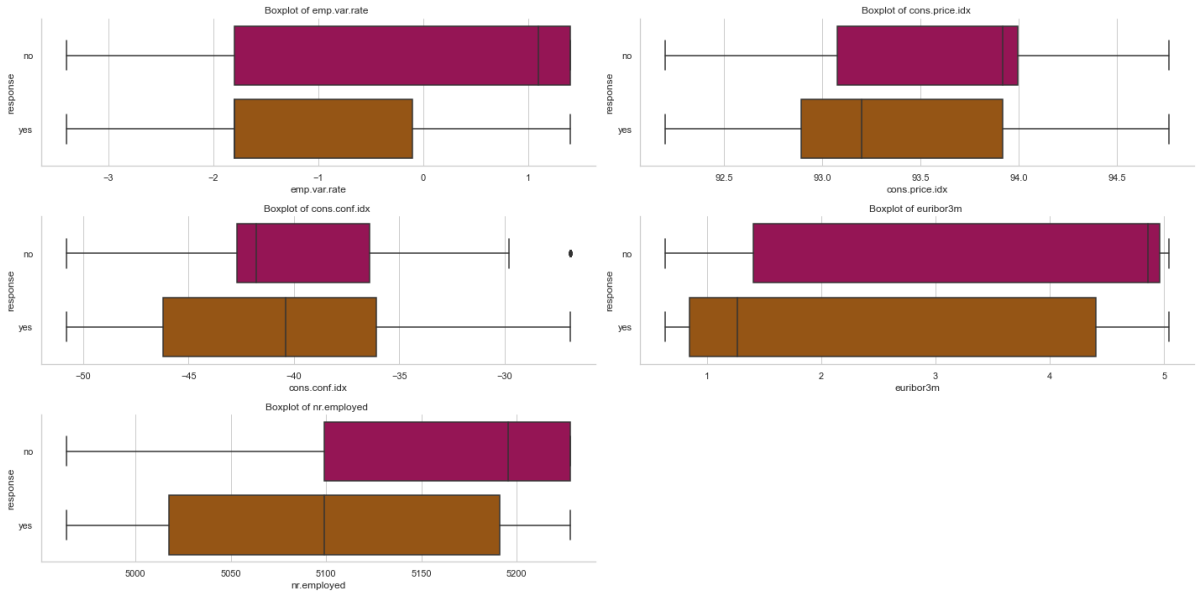
Observation

As we can see from the plot, “yes” clients and “no” clients are forming two relatively separate clusters. Compared to “no” clients”, “yes” clients were contacted by fewer times and had longer call duration. More importantly, after five campaign calls, clients are more likely to reject the term deposit unless the duration is high. Most “yes” clients were approached by less than 10 times.

Insights

This suggests that the bank should resist calling a client for more than six times, which can be disturbing and increase dissatisfaction.

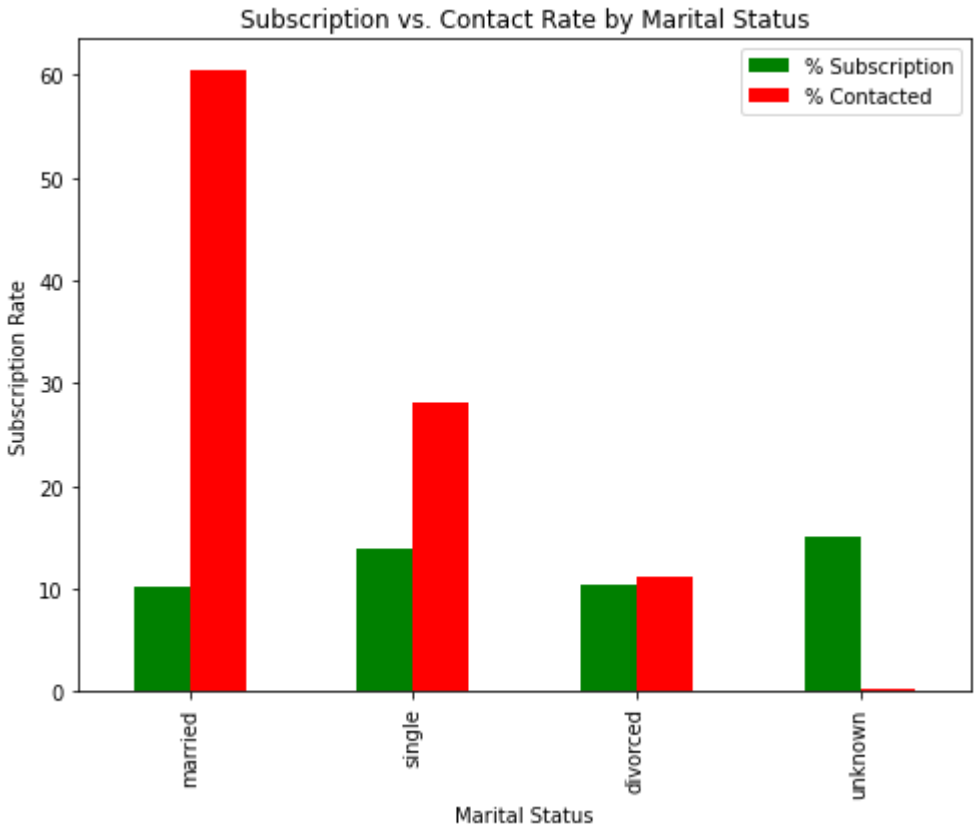
In [169]:
```
#Distribution of social and economic context attributes
plt.figure(figsize=[20,10])
socio_eco_variables=["emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed"]
n=1
for i in socio_eco_variables:
    plt.subplot(3,2,n)
    sns.boxplot(x=i, data=df, y='response', palette = 'brg')
    sns.despine()
    plt.title("Boxplot of {} ".format(i))
    n=n+1
plt.tight_layout()
plt.show()
```



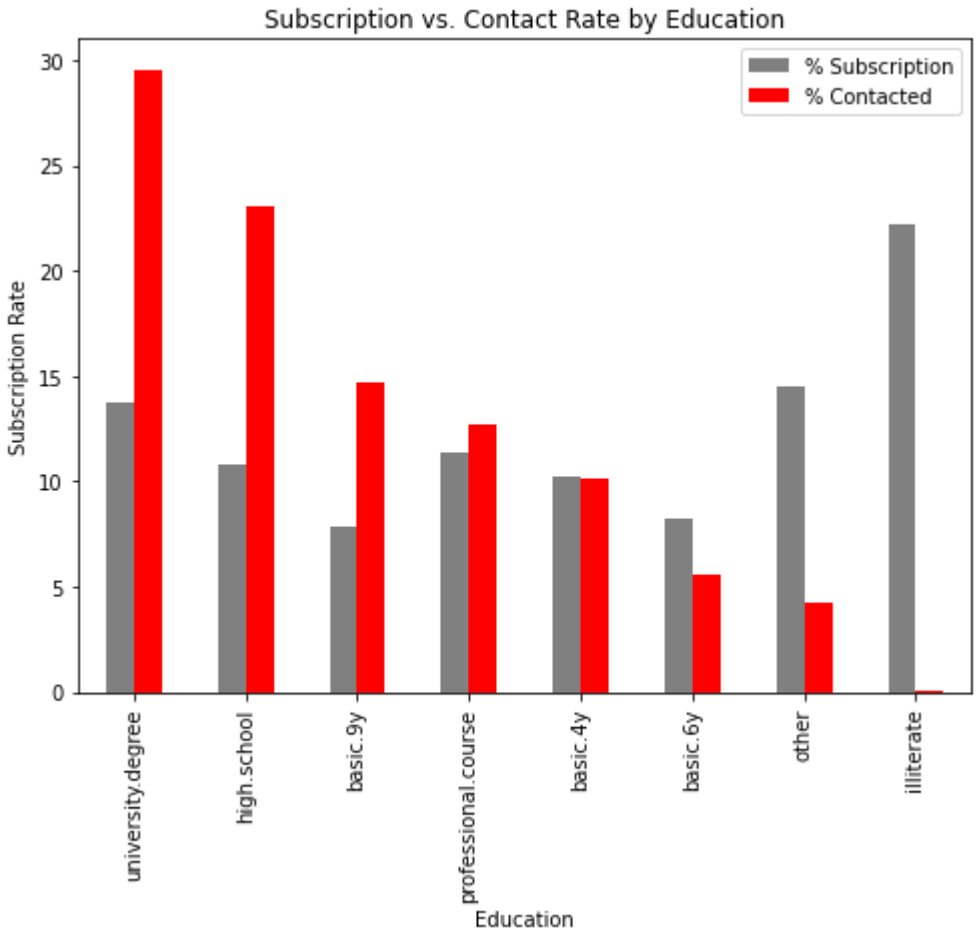
In [45]:
```
cnt_marital_resp = pd.crosstab(df['response'],df['marital']).apply(lambda x: x/x.sum() * 100)
cnt_marital_resp = cnt_marital_resp.transpose()
```

In [44]:
```
g_marital = pd.DataFrame(df['marital'].value_counts())
g_marital['% Contacted'] = g_marital['marital']*100/g_marital['marital'].sum()
g_marital['% Subscription'] = cnt_marital_resp['yes']
```

In [42]:
```
plot_marital = g_marital[['% Subscription','% Contacted']].plot(kind = 'bar',
                                                    figsize=(8,6), color = ('green',
'red'))
plt.xlabel('Marital Status')
plt.ylabel('Subscription Rate')
plt.title('Subscription vs. Contact Rate by Marital Status')
plt.show()
```



In [47]:
```
cnt_edu_resp = pd.crosstab(df['response'],df['education']).apply(lambda x: x/x
.sum() * 100)
cnt_edu_resp = cnt_edu_resp.transpose()


g_edu = pd.DataFrame(df['education'].value_counts())
g_edu['% Contacted'] = g_edu['education']*100/g_edu['education'].sum()
g_edu['% Subscription'] = cnt_edu_resp['yes']


plot_edu = g_edu[['% Subscription','% Contacted']].plot(kind = 'bar',
                                                        figsize=(8,6), color = ('grey',
'red'))
plt.xlabel('Education')
plt.ylabel('Subscription Rate')
plt.title('Subscription vs. Contact Rate by Education')
plt.show()
```



Observations:

In the education-response graph, university degree, professional course and unknown have higher conversion.

In marital – response graph, single clients followed by married have high conversion rate. Divorced clients show lowest conversion.

Insights:

Target clients with higher level education, need to find what unknown indicate.

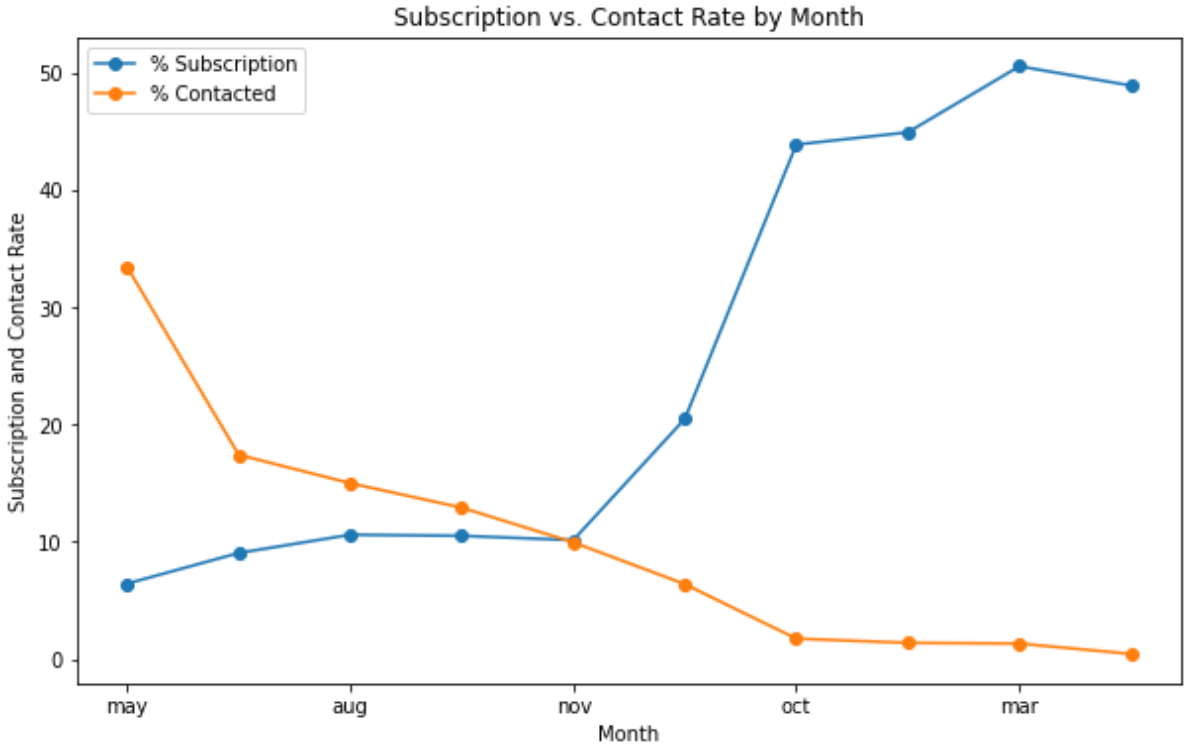
Target clients who are single, with second priority married client

In [51]:
```
cnt_month_resp = pd.crosstab(df['response'], df['month']).apply(lambda x: x/x.sum() * 100)
cnt_month_resp = cnt_month_resp.transpose()
```

In [56]:
```
month_df = pd.DataFrame(df['month'].value_counts())
month_df['% Contacted'] = month_df['month']*100/month_df['month'].sum()
month_df['% Subscription'] = cnt_month_resp['yes']
```

In [57]:
```
g_month = month_df[['% Subscription','% Contacted']].plot(kind ='line',figsize = (10,6), marker = 'o')

plt.title('Subscription vs. Contact Rate by Month')
plt.ylabel('Subscription and Contact Rate')
plt.xlabel('Month')
```

Out[57]: Text(0.5, 0, 'Month')



Insights:

Initiate the telemarketing campaign in fall or spring

Besides customer characteristics, external factors may also have an impact on the subscription rate, such as seasons and the time of calling. So the month of contact is also analyzed here.

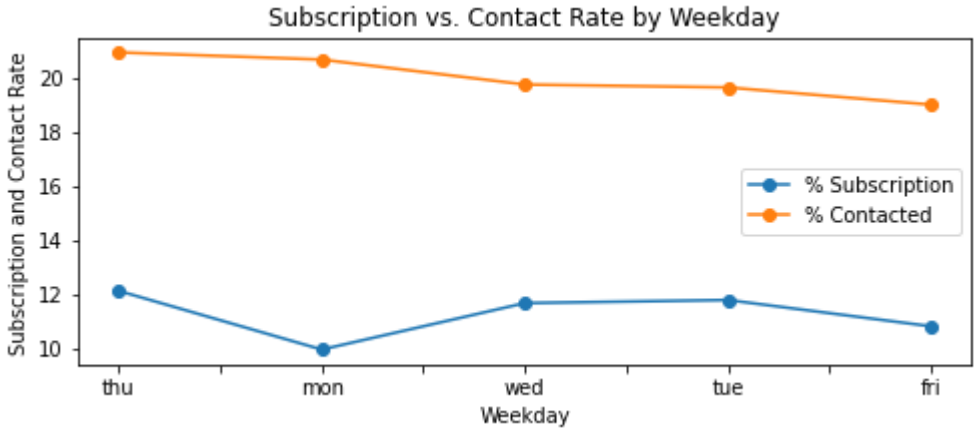
This line chart displays the bank’s contact rate in each month as well as clients’ response rate in each month. One way to evaluate the effectiveness of the bank's marketing plan is to see whether these two lines have a similar trend over the same time horizon.

The bank contacted most clients between May and August. The highest contact rate is around 30%, which happened in May, while the contact rate is closer to 0 in March, September, October, and December. However, the subscription rate showed a different trend. The highest subscription rate occurred in March, which is over 50%, and all subscription rates in September, October, and December are over 40%. Clearly, these two lines move in different directions which strongly indicates the inappropriate timing of the bank’s marketing campaign. To improve the marketing campaign, the bank should consider initiating the telemarketing campaign in fall and spring when the subscription rate tends to be higher.

Nevertheless, the bank should be cautious when analyzing external factors. More data from previous marketing campaign should be collected and analyzed to make sure that this seasonal effect is constant over time and applicable to the future.

In [5]:
```
cnt_week_resp = pd.crosstab(df['response'], df['day_of_week']).apply(lambda x:
x/x.sum() * 100)
cnt_week_resp = cnt_week_resp.transpose()

week_df = pd.DataFrame(df['day_of_week'].value_counts())
week_df['% Contacted'] = week_df['day_of_week']*100/week_df['day_of_week'].sum
()
week_df['% Subscription'] = cnt_week_resp['yes']


g_week = week_df[['% Subscription','% Contacted']].plot(kind ='line',figsize =
(8,3), marker = 'o')

plt.title('Subscription vs. Contact Rate by Weekday')
plt.ylabel('Subscription and Contact Rate')
plt.xlabel('Weekday')
```

Out[5]: Text(0.5, 0, 'Weekday')



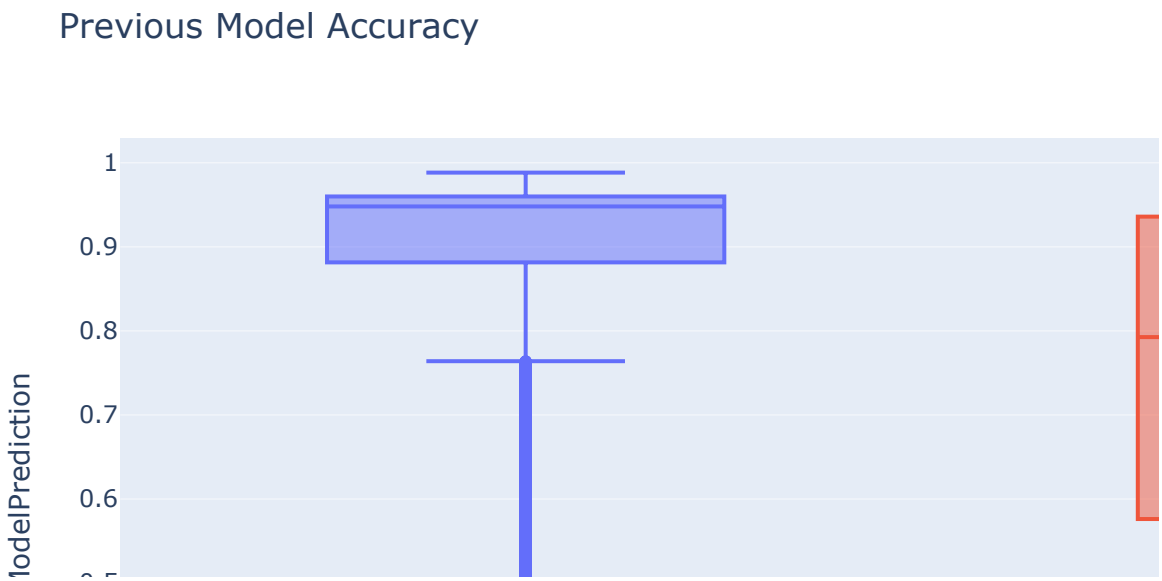
Observation

Contacts per day chart displays response on Tuesday and Wednesday is more as compared to Monday where its low even when contact made are max of week.

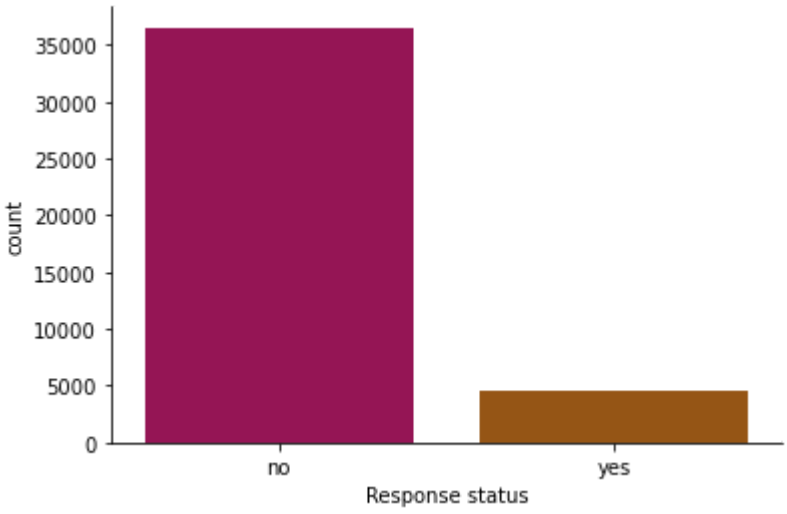
Insights:

Contacts should be made preferably on Tuesday, Wednesday and Thursday. Productivity of marketing team is maximum on Monday and least on Friday.

In [15]:
```
fig = px.box(df, y= 'ModelPrediction', x="response", title = "Previous Model Accuracy", color = 'response')
fig.show()
```



In [16]:
```
g = sns.countplot(data = df, x = 'response', palette = 'brg')
g.set_xlabel('Response status')
sns.despine()
```



On the diagram we see that counts for 'yes' and 'no' values for 'response' are close, so we can use accuracy as a metric for a model, which predicts the campaign outcome. But in our case they are not close

For Response: No The interquartile range here is 88% to 96% For Response: Yes The interquartile range here is 94% to 57%

Majority of the responses received were negative and the response column shows a higher possibility of positive response for FD when the real life its negative. The model needs to be changed.

In [ ]: