In [1]:

**from** **IPython.display** **import** HTML

In [2]:

HTML('''<script>  
code\_show=true;   
function code\_toggle() {  
 if (code\_show){  
 $('div.input').hide();  
 } else {  
 $('div.input').show();  
 }  
 code\_show = !code\_show  
}   
$( document ).ready(code\_toggle);  
</script>  
The raw code for this IPython notebook is by default hidden for easier reading.  
To toggle on/off the raw code, click <a href="javascript:code\_toggle()">here</a>.''')

Out[2]:

The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

**Introduction**[**¶**](#gjdgxs)

**TeleMarketing** was introduced in 1970s as a method of direct marketing. Telemarketing is a method where leads are generated through telephone calls or Face to Face Web-Conference. Charitable organizations, alumni associations, and political parties often use telemarketing to solicit donations. Marketing research companies use telemarketing techniques to survey the prospective or past customers of a client’s business in order to assess market acceptance of or satisfaction with a particular product, service, brand, or company. Public opinion polls are conducted in a similar manner.

This project will deal with a database that provides information which can help in analysing whether the caller will be a potential buyer or not. The Database contains the variables as of a Portuguese bank that wants to sell 'Term Loans' to it's potential callers.

The Variables contained in the database are listed as follows:

age - Age of the client- (numeric)

job - Client’s occupation - (categorical) (admin, bluecollar, entrepreneur, housemaid, management, retired, selfemployed, services, student, technician, unemployed, unknown)

marital - Client’s marital status - (categorical) (divorced, married, single, unknown, note: divorced means divorced or widowed)

education - Client’s education level - (categorical) (basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown)

default - Indicates if the client has credit in default - (categorical) (no, yes, unknown)

housing - Does the client has a housing loan? - (categorical) (no, yes, unknown)

loan - Does the client has a personal loan? - (categorical) (no, yes, unknown’)

contact - Type of communication contact - (categorical) (cellular, telephone)

month - Month of last contact with client - (categorical) (January - December)

day\_of\_week - Day of last contact with client - (categorical) (Monday - Friday)

duration - Duration of last contact with client, in seconds - (numeric) For benchmark purposes only, and not reliable for predictive modeling

campaign - Number of client contacts during this campaign - (numeric) (includes last contact)

pdays - Number of days from last contacted from a previous campaign - (numeric) (999 means client was not previously contacted)

previous - Number of client contacts performed before this campaign - (numeric)

poutcome - Previous marketing campaign outcome - (categorical) (failure, nonexistent , success)

emp.var.rate - Quarterly employment variation rate - (numeric)

cons.price.idx - Monthly consumer price index - (numeric)

cons.conf.idx - Monthly consumer confidence index - (numeric)

euribor3m - Daily euribor 3 month rate - (numeric)

nr.employed - Quarterly number of employees - (numeric)

Output variable (desired target) - Term Deposit - subscription verified (binary: ‘yes’,‘no’)

**Data Analysis & Findings**[**¶**](#30j0zll)

**1.Import the necessary libraries in Python.**

In [1]:

**import** **numpy** **as** **np**  
**import** **pandas** **as** **pd**  
**import** **matplotlib.pyplot** **as** **plt**  
**import** **seaborn** **as** **sns**  
%**matplotlib** inline

**The First Four values of the Data Frame**

In [2]:

data = pd.read\_csv('train.csv',sep=';')

In [3]:

data.head(4)

Out[3]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **job** | **marital** | **education** | **default** | **housing** | **loan** | **contact** | **month** | **day\_of\_week** | **...** | **campaign** | **pdays** | **previous** | **poutcome** | **emp.var.rate** | **cons.price.idx** | **cons.conf.idx** | **euribor3m** | **nr.employed** | **y** |
| **0** | 30 | blue-collar | married | basic.9y | no | yes | no | cellular | may | fri | ... | 2 | 999 | 0 | nonexistent | -1.8 | 92.893 | -46.2 | 1.313 | 5099.1 | no |
| **1** | 39 | services | single | high.school | no | no | no | telephone | may | fri | ... | 4 | 999 | 0 | nonexistent | 1.1 | 93.994 | -36.4 | 4.855 | 5191.0 | no |
| **2** | 25 | services | married | high.school | no | yes | no | telephone | jun | wed | ... | 1 | 999 | 0 | nonexistent | 1.4 | 94.465 | -41.8 | 4.962 | 5228.1 | no |
| **3** | 38 | services | married | basic.9y | no | unknown | unknown | telephone | jun | fri | ... | 3 | 999 | 0 | nonexistent | 1.4 | 94.465 | -41.8 | 4.959 | 5228.1 | no |

4 rows × 21 columns

**Some Technical Information about the Data Frame**

In [4]:

data.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 2999 entries, 0 to 2998  
Data columns (total 21 columns):  
age 2999 non-null int64  
job 2999 non-null object  
marital 2999 non-null object  
education 2999 non-null object  
default 2999 non-null object  
housing 2999 non-null object  
loan 2999 non-null object  
contact 2999 non-null object  
month 2999 non-null object  
day\_of\_week 2999 non-null object  
duration 2999 non-null int64  
campaign 2999 non-null int64  
pdays 2999 non-null int64  
previous 2999 non-null int64  
poutcome 2999 non-null object  
emp.var.rate 2999 non-null float64  
cons.price.idx 2999 non-null float64  
cons.conf.idx 2999 non-null float64  
euribor3m 2999 non-null float64  
nr.employed 2999 non-null float64  
y 2999 non-null object  
dtypes: float64(5), int64(5), object(11)  
memory usage: 492.1+ KB

There are 11 categorical variables and 10 continuous variables.

**Quantitative Summary**

In [5]:

data.describe().T

Out[5]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| **age** | 2999.0 | 39.889296 | 10.282229 | 18.000 | 32.000 | 38.000 | 47.000 | 88.000 |
| **duration** | 2999.0 | 255.085028 | 256.265737 | 0.000 | 101.000 | 180.000 | 316.000 | 3643.000 |
| **campaign** | 2999.0 | 2.526842 | 2.556995 | 1.000 | 1.000 | 2.000 | 3.000 | 35.000 |
| **pdays** | 2999.0 | 962.227743 | 187.599866 | 0.000 | 999.000 | 999.000 | 999.000 | 999.000 |
| **previous** | 2999.0 | 0.192064 | 0.551960 | 0.000 | 0.000 | 0.000 | 0.000 | 6.000 |
| **emp.var.rate** | 2999.0 | 0.079093 | 1.559126 | -3.400 | -1.800 | 1.100 | 1.400 | 1.400 |
| **cons.price.idx** | 2999.0 | 93.584553 | 0.581162 | 92.201 | 93.075 | 93.876 | 93.994 | 94.767 |
| **cons.conf.idx** | 2999.0 | -40.578226 | 4.594237 | -50.800 | -42.700 | -41.800 | -36.400 | -26.900 |
| **euribor3m** | 2999.0 | 3.605769 | 1.739625 | 0.635 | 1.334 | 4.857 | 4.961 | 5.045 |
| **nr.employed** | 2999.0 | 5165.642314 | 74.384632 | 4963.600 | 5099.100 | 5191.000 | 5228.100 | 5228.100 |

**Y - Will take the term loan or not!**[**¶**](#1fob9te)

This variable tells whether the call led to a successful subscription of Term Deposit or not.

In [6]:

data\_no = data[data['y']=='no']  
data\_yes = data[data['y']=='yes']

In [7]:

sns.countplot(data['y'])  
plt.tight\_layout()  
plt.show()

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1460: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data)

In [8]:

print('VALUES IN PERCENTAGE')  
print(data[data['y']=='yes']['y'].value\_counts()/len(data['y'])\*100)  
print(data[data['y']=='no']['y'].value\_counts()/len(data['y'])\*100)

VALUES IN PERCENTAGE  
yes 11.037012  
Name: y, dtype: float64  
no 88.962988  
Name: y, dtype: float64

Here, **12%** of the calls led to a successful subscription while **88%** of the calls did not lead to a subscription.

**Short Summary**[**¶**](#3znysh7)

In [9]:

**def** Graph\_Summary(input\_value):  
 **if**((type(data[input\_value].iloc[0]) == np.int64) | (type(data[input\_value].iloc[0]) == np.float64) ) :  
   
 f = plt.figure(figsize=(15,15))  
 f.set\_figheight(8)  
 f.set\_figwidth(25)  
  
 plt.subplot(131)  
 sns.set()  
 plt.hist(data[input\_value],bins = 50)  
 plt.xlabel(input\_value)  
 plt.ylabel('F(X)')  
   
  
  
 plt.subplot(132)  
 sns.set()  
 X = np.sort(data[input\_value])  
 Y = np.arange(1,len(X)+1)/len(X)  
 plt.plot(X,Y,marker='.',linestyle='none')  
 plt.xlabel(input\_value)  
 plt.ylabel('ECDF')  
  
  
  
 plt.subplot(133)  
 sns.set()  
 plt.boxplot(data[input\_value])  
 plt.xlabel(input\_value)  
   
 plt.show()  
 **else**: **return** **None**

In [10]:

**for** i **in** data.columns:  
 Graph\_Summary(i)

The above graphs shows the statistical summary of continuous variables.

In [11]:

**def** Cat\_Summary(input\_value):  
   
 **if**((type(data[input\_value].iloc[0]) != np.int64) | (type(data[input\_value].iloc[0]) != np.float64) ) :  
 **if**(data[input\_value].nunique() < 10 ):  
 print(input\_value)  
 print('Number of Uniques: ',data[input\_value].nunique(),' ','Unique Values: ',data[input\_value].unique())  
 print('------------------------------------------------------------------------------------------------------')  
 **else**: **return** **None**   
   
 **else**: **return** **None**

In [12]:

print('Categorical Unique Values:')  
**for** i **in** data.columns:  
 Cat\_Summary(i)

Categorical Unique Values:  
marital  
Number of Uniques: 4 Unique Values: ['married' 'single' 'divorced' 'unknown']  
------------------------------------------------------------------------------------------------------  
education  
Number of Uniques: 7 Unique Values: ['basic.9y' 'high.school' 'university.degree' 'professional.course'  
 'basic.6y' 'basic.4y' 'unknown']  
------------------------------------------------------------------------------------------------------  
default  
Number of Uniques: 2 Unique Values: ['no' 'unknown']  
------------------------------------------------------------------------------------------------------  
housing  
Number of Uniques: 3 Unique Values: ['yes' 'no' 'unknown']  
------------------------------------------------------------------------------------------------------  
loan  
Number of Uniques: 3 Unique Values: ['no' 'unknown' 'yes']  
------------------------------------------------------------------------------------------------------  
contact  
Number of Uniques: 2 Unique Values: ['cellular' 'telephone']  
------------------------------------------------------------------------------------------------------  
day\_of\_week  
Number of Uniques: 5 Unique Values: ['fri' 'wed' 'mon' 'thu' 'tue']  
------------------------------------------------------------------------------------------------------  
previous  
Number of Uniques: 7 Unique Values: [0 2 1 3 5 4 6]  
------------------------------------------------------------------------------------------------------  
poutcome  
Number of Uniques: 3 Unique Values: ['nonexistent' 'failure' 'success']  
------------------------------------------------------------------------------------------------------  
y  
Number of Uniques: 2 Unique Values: ['no' 'yes']  
------------------------------------------------------------------------------------------------------

**Conclusion**[**¶**](#2et92p0)

From the graphical summary of the **continuous varaibles**, it was observed that there existed a difference in mean and median, although the boxplot does not show any outliers, but this difference tells us that outliers are existing in these variables. Outliers, here, would be capped or floored rather than removed.

From the **categorical summary** it was observed that there existed some unknown categories. Such are to be treated as NAs and removed from the dataset.

**Capping Outliers**[**¶**](#tyjcwt)

**Duration**[**¶**](#3dy6vkm)

In [13]:

f = plt.figure(figsize = (15,8))  
f.set\_figheight(8)  
f.set\_figwidth(15)  
   
plt.subplot(121)  
sns.distplot(data['duration'])  
plt.subplot(122)  
plt.boxplot(data['duration'])  
plt.xlabel('Box Plot Of Duration Variable')

Out[13]:

Text(0.5,0,'Box Plot Of Duration Variable')

In [14]:

f = plt.figure(figsize = (15,8))  
f.set\_figheight(8)  
f.set\_figwidth(15)  
   
plt.subplot(121)  
sns.set()  
X = np.sort(data\_no['duration'])  
Y = np.arange(1,len(X)+1)/len(X)  
plt.plot(X,Y,marker='.',linestyle='none')  
plt.ylabel('ECDF')  
plt.xlabel('Duration for Calls Which did not sell Term Loans')  
plt.subplot(122)  
sns.set()  
X = np.sort(data\_yes['duration'])  
Y = np.arange(1,len(X)+1)/len(X)  
plt.plot(X,Y,marker='.',linestyle='none')  
plt.ylabel('ECDF')  
plt.xlabel('Duration for Calls Which sold Term Loans')

Out[14]:

Text(0.5,0,'Duration for Calls Which sold Term Loans')

**The graph above clearly shows that 100% of the data is evenly distributed upto the duration of 1500 units. All the other calls greater than 1500 units are unique and rare. These should be positively capped to avoid error in data probability.**

In [15]:

**def** cap\_outs(values):  
 **for** i **in** range(0,len(data)-1):  
 **if**(values > 1500):  
 values = 1500  
 **else**: **return** values  
 **return** values

In [16]:

data['duration']=data['duration'].apply(cap\_outs)

**PDAYS**[**¶**](#1t3h5sf)

In [17]:

plt.figure(figsize = (15,8))  
sns.countplot(data['pdays'])

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1460: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data)

Out[17]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x52e7ea1198>

**The no. of days till last contacted in previous campaign shows that there are calls which were last contacted 900 days ago. This carries no value and treated the same as 0 days till last contacted. Thus we will make the days greater than 100 to 0.**

In [18]:

**def** cap\_outs(values):  
 **for** i **in** range(0,len(data)-1):  
 **if**(values > 900):  
 values = 0  
 **else**: **return** values  
 **return** values  
  
data['pdays']=data['pdays'].apply(cap\_outs)

**Filtering Out the Unknown Values**[**¶**](#4d34og8)

In [19]:

print('Categorical Unique Values:')  
**for** i **in** data.columns:  
 Cat\_Summary(i)

Categorical Unique Values:  
marital  
Number of Uniques: 4 Unique Values: ['married' 'single' 'divorced' 'unknown']  
------------------------------------------------------------------------------------------------------  
education  
Number of Uniques: 7 Unique Values: ['basic.9y' 'high.school' 'university.degree' 'professional.course'  
 'basic.6y' 'basic.4y' 'unknown']  
------------------------------------------------------------------------------------------------------  
default  
Number of Uniques: 2 Unique Values: ['no' 'unknown']  
------------------------------------------------------------------------------------------------------  
housing  
Number of Uniques: 3 Unique Values: ['yes' 'no' 'unknown']  
------------------------------------------------------------------------------------------------------  
loan  
Number of Uniques: 3 Unique Values: ['no' 'unknown' 'yes']  
------------------------------------------------------------------------------------------------------  
contact  
Number of Uniques: 2 Unique Values: ['cellular' 'telephone']  
------------------------------------------------------------------------------------------------------  
day\_of\_week  
Number of Uniques: 5 Unique Values: ['fri' 'wed' 'mon' 'thu' 'tue']  
------------------------------------------------------------------------------------------------------  
previous  
Number of Uniques: 7 Unique Values: [0 2 1 3 5 4 6]  
------------------------------------------------------------------------------------------------------  
poutcome  
Number of Uniques: 3 Unique Values: ['nonexistent' 'failure' 'success']  
------------------------------------------------------------------------------------------------------  
y  
Number of Uniques: 2 Unique Values: ['no' 'yes']  
------------------------------------------------------------------------------------------------------

In [20]:

**for** i **in** range(0,len(data.columns)-1):  
 **for** j **in** range(0,len(data)-1):  
 **if**(data.iloc[j,i] == 'unknown'):  
 data.iloc[j,i] = **None**  
 **else**: **None**

In [21]:

data.isna().sum()

Out[21]:

age 0  
job 32  
marital 9  
education 138  
default 596  
housing 77  
loan 77  
contact 0  
month 0  
day\_of\_week 0  
duration 0  
campaign 0  
pdays 0  
previous 0  
poutcome 0  
emp.var.rate 0  
cons.price.idx 0  
cons.conf.idx 0  
euribor3m 0  
nr.employed 0  
y 0  
dtype: int64

In [22]:

len(data)

Out[22]:

2999

In [23]:

data = data.dropna()

In [24]:

data.isna().sum()

Out[24]:

age 0  
job 0  
marital 0  
education 0  
default 0  
housing 0  
loan 0  
contact 0  
month 0  
day\_of\_week 0  
duration 0  
campaign 0  
pdays 0  
previous 0  
poutcome 0  
emp.var.rate 0  
cons.price.idx 0  
cons.conf.idx 0  
euribor3m 0  
nr.employed 0  
y 0  
dtype: int64

In [25]:

len(data)

Out[25]:

2226

**Exploratory Data Analysis**[**¶**](#2s8eyo1)

**Figure 1:**

In [26]:

plt.figure(figsize=(12,10))  
sns.boxplot(y=data['age'],x=data['job'])  
plt.tight\_layout()  
plt.show()

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:462: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 box\_data = remove\_na(group\_data)

**In Figure 1:**

**This Box Plot shows the median ages for different Job status and the median age for each job status is given below:**

In [27]:

data.groupby('job')['age'].median()

Out[27]:

job  
admin. 35.0  
blue-collar 35.0  
entrepreneur 41.0  
housemaid 43.0  
management 40.0  
retired 58.0  
self-employed 37.0  
services 36.0  
student 26.0  
technician 36.0  
unemployed 36.5  
Name: age, dtype: float64

**Figure 2**

In [28]:

**import** **matplotlib**  
f = plt.figure(figsize=(8,8))  
f.set\_figheight(10)  
f.set\_figwidth(10)  
   
plt.subplot(221)  
plt.pie(list(data['housing'].value\_counts()),labels=['Yes','No'],autopct='**%1.1f%%**',)  
plt.xlabel('Housing',{'fontsize':15})  
  
plt.subplot(222)  
plt.pie(list(data['loan'].value\_counts()),labels=['Yes','No'],autopct='**%1.1f%%**')  
plt.xlabel('Loans',{'fontsize':15})  
  
plt.subplot(223)  
sns.countplot(data['housing'])  
plt.xlabel('Housing',{'fontsize':15})  
  
plt.subplot(224)  
sns.countplot(data['loan'])  
plt.xlabel('Loans',{'fontsize':15})  
matplotlib.rcParams.update({'font.size': 15})

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1460: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data)

**In Figure 2**,

**Pie Chart Shows that:**

1)Out of the total, **55.1%** has housing loans and **44.9%** do not have a Housing Loan.

2)Again, for personal loans, **83.7%** did have personal loans while the rest **16.3%** do not have personal loan.

**The Count Plot shows that:**

1) 1227 of them has house loan and 999 of them do not have house loan.

2) 1863 of them has personal loan and 363 of them do not have personal loan.

**Figure 3**

In [29]:

f = plt.figure(figsize=(8,8))  
f.set\_figheight(13)  
f.set\_figwidth(20)  
   
plt.subplot(231)  
plt.pie(list(data['education'].value\_counts()),labels=data['education'].unique(),autopct='**%1.1f%%**',)  
plt.xlabel('education',{'fontsize':15})  
  
plt.subplot(232)  
plt.pie(list(data['default'].value\_counts()),labels=data['default'].unique(),autopct='**%1.1f%%**')  
plt.xlabel('default',{'fontsize':15})  
  
plt.subplot(233)  
plt.pie(list(data['contact'].value\_counts()),labels=data['contact'].unique(),autopct='**%1.1f%%**')  
plt.xlabel('contact',{'fontsize':15})  
  
plt.subplot(234)  
plt.pie(list(data['poutcome'].value\_counts()),labels=data['poutcome'].unique(),autopct='**%1.1f%%**')  
plt.xlabel('poutcome',{'fontsize':15})

Out[29]:

Text(0.5,0,'poutcome')

**In Figure 3:**

**The Pie Chart Shows:**

1) Percentage of Education, where the less educated are contacted more because these are the customers who will require the term loan more than the others.

2) Percentage of default shows that all those who have been contacted did not have a defaulting history or did not default in paying back of any loans.

3) Percentage of Contact, which shows 68.6% of the customers has a cell phone while the rest 31.4% has a telephone.

4) Percentage of POutcome, which shows that in the previous campaign, there was 4% success for buying a term loan, 11.5% failure and 84.5% non-existent(Maybe they said they would think about it and was never contacted again).

**Out of calls that were success in the previous campaign, how many of them took the term loan in this campaign?**

In [30]:

success = data[data['poutcome']=='success']

In [31]:

plt.figure(figsize = (7,7))  
plt.pie(list(success['y'].value\_counts()),labels=success['y'].unique(),autopct='**%1.1f%%**')  
plt.xlabel("Percentage of term loan taken or not compared to previous campaign's success",{'fontsize':15})  
plt.legend()

Out[31]:

<matplotlib.legend.Legend at 0x52e726cba8>

In [32]:

**import** **matplotlib**  
f = plt.figure(figsize=(8,8))  
f.set\_figheight(13)  
f.set\_figwidth(13)  
   
plt.subplot(221)  
plt.pie(list(data['marital'].value\_counts()),labels=['single', 'married', 'divorced'],autopct='**%1.1f%%**')  
plt.xlabel('1 Marital Status')  
  
plt.subplot(222)  
sns.countplot(data['marital'],hue=data['housing'])  
plt.xlabel('2 Marital status and House Ownership')  
  
plt.subplot(223)  
sns.countplot(data['marital'],hue=data['loan'])  
plt.xlabel('3 Marital status and Personal Loan',{'fontsize':15})  
  
plt.subplot(224)  
sns.countplot(data['marital'],hue=data['y'])  
plt.xlabel('4 Marital status and Subscribed to Term Loan',{'fontsize':15})  
  
plt.tight\_layout()

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1508: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data[hue\_mask])

In [33]:

plt.figure(figsize = (12,7))  
sns.heatmap(data.corr())

Out[33]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x52e7f6bc18>

**Monthly Trends of Continuous Variables**

In [34]:

month = {'apr':4,'aug':8,'dec':12,'jul':7,'jun':6,'mar':3,'may':5,'nov':11,'oct':10,'sep':9}  
data['month'] = data['month'].map(month)

In [35]:

by\_month = data.groupby('month').sum()

In [36]:

plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.subplot(4,3,1)  
plt.plot(by\_month['emp.var.rate'],'r-')  
plt.xlabel('Employee Variation Rate by Months')  
plt.subplot(4,3,2)  
plt.plot(np.log(by\_month['cons.price.idx']),'b-')  
plt.xlabel('Consumer Price Index by Months')  
plt.subplot(4,3,3)  
plt.plot(np.log(by\_month['euribor3m']),'g-')  
plt.xlabel('Europe Inter Bank Offered Rate')  
  
plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.margins(0.1)  
plt.subplot(4,3,4)  
sns.barplot(y=data['emp.var.rate'],x=data['month'])  
plt.xlabel('Employee Variation Rate by Months')  
plt.subplot(4,3,5)  
sns.barplot(y=np.log(data['cons.price.idx']),x=data['month'])  
plt.xlabel('Consumer Price Index by Months')  
plt.subplot(4,3,6)  
sns.barplot(y=np.log(data['euribor3m']),x=data['month'])  
plt.xlabel('Europe Inter Bank Offered Rate')  
  
plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.subplot(4,3,7)  
plt.plot(by\_month['duration'],'r-')  
plt.xlabel('duration')  
plt.subplot(4,3,8)  
plt.plot(by\_month['campaign'],'b-')  
plt.xlabel('campaign')  
plt.subplot(4,3,9)  
plt.plot(by\_month['previous'],'g-')  
plt.xlabel('previous')  
  
plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.margins(0.1)  
plt.subplot(4,3,10)  
sns.barplot(y=data['duration'],x=data['month'])  
plt.xlabel('duration')  
plt.subplot(4,3,11)  
sns.barplot(y=data['campaign'],x=data['month'])  
plt.xlabel('campaign')  
plt.subplot(4,3,12)  
sns.barplot(y=data['previous'],x=data['month'])  
plt.xlabel('previous')

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1460: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data)

Out[36]:

Text(0.5,0,'previous')

**Weekday's Trends of Continuous Variables**

In [37]:

dow = {'mon':1,'tue':2,'wed':3,'thu':4,'fri':5}  
data['day\_of\_week'] = data['day\_of\_week'].map(dow)

In [38]:

by\_dow = data.groupby('day\_of\_week').sum()

In [39]:

plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.subplot(4,3,1)  
plt.plot(by\_dow['emp.var.rate'],'r-')  
plt.xlabel('Employee Variation Rate by day\_of\_week')  
plt.subplot(4,3,2)  
plt.plot(by\_dow['cons.price.idx'],'b-')  
plt.xlabel('Consumer Price Index by day\_of\_week')  
plt.subplot(4,3,3)  
plt.plot(by\_dow['euribor3m'],'g-')  
plt.xlabel('Europe Inter Bank Offered Rate')  
  
plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.margins(0.1)  
plt.subplot(4,3,4)  
sns.barplot(y=data['emp.var.rate'],x=data['day\_of\_week'])  
plt.xlabel('Employee Variation Rate by day\_of\_week')  
plt.subplot(4,3,5)  
sns.barplot(y=data['cons.price.idx'],x=data['day\_of\_week'])  
plt.xlabel('Consumer Price Index by day\_of\_week')  
plt.subplot(4,3,6)  
sns.barplot(y=data['euribor3m'],x=data['day\_of\_week'])  
plt.xlabel('Europe Inter Bank Offered Rate by day\_of\_week')  
  
plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.subplot(4,3,7)  
plt.plot(by\_dow['duration'],'r-')  
plt.xlabel('duration')  
plt.subplot(4,3,8)  
plt.plot(by\_dow['campaign'],'b-')  
plt.xlabel('campaign')  
plt.subplot(4,3,9)  
plt.plot(by\_dow['previous'],'g-')  
plt.xlabel('previous')  
  
plt.figure(figsize=(15,10))  
f.set\_figheight(10)  
f.set\_figwidth(25)  
  
plt.margins(0.1)  
plt.subplot(4,3,10)  
sns.barplot(y=data['duration'],x=data['day\_of\_week'])  
plt.xlabel('duration')  
plt.subplot(4,3,11)  
sns.barplot(y=data['campaign'],x=data['day\_of\_week'])  
plt.xlabel('campaign')  
plt.subplot(4,3,12)  
sns.barplot(y=data['previous'],x=data['day\_of\_week'])  
plt.xlabel('previous')

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1460: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data)

Out[39]:

Text(0.5,0,'previous')