# Assignment 2: Data Visualization and Cleaning

**Submitted on:** March 14, 2021

**Submitted to :** Sasha Ali-Hosein

## Summary

In this assignment we have worked on the following areas:
1- We have analyzed the data properly using bivariate and univariate analysis.
2- We have calculated the correlations between the features.
3- We have created new features using one hot coding and label encoding.
4- We have checked the features with box plot and then have imputed mean value in the age and duration column for the outlier removal
5- We have calculated the correlation of all the features and have shortlisted top correlated features for modeling.

### ***Feature Engineering***

When doing Feature Engineering, we aimed to create more features to find the possible relationship between categorical columns. It helps us to deep dive into the banking data set which has variables making the highest impact in our prediction. During Featuring Engineering we find the correlation between each feature and then take the highest correlated features for further analysis.

In the data set we have several important features such as,

- default_no
- loan_unknown
- housing_unknown
- default_unknown
- emp.var.rate
- nr.employed
- loan_no
- loan_yes
- marital_single
- duration
- euribor3m
- cons.conf.idx
- previous
- pdays
- housing_yes
- housing_no
- contact

- job_technician
- Education_professional.course
- marital_married

In order to properly analyze the data we used the concept of one hot encoding and binning on categorical variables and numerical variables respectively. We were able to treat our categorical variables so they can be ready for further analysis such as the job feature was divided into further features such as job_admin, job_technician etc and we used one hot encoding to assign values to these new features.

## *Details*

In this part, we performed exploratory data analysis to gather patterns and insights that can help our client to attract more customers for its 'term deposit' campaign, to spend marketing dollars wisely and to extract useful insights.

## *Data Exploration*

This section outlines the process followed in obtaining the data, initial setup and understanding of the data.
The uci machine learning repository was used to download the banking data for analysis. The repository has 4 data files. However, bank-additional-full.csv with 41188 observations and 20 inputs is selected to upload in Jupyter notebook for further analysis. The data is already cleaned, at least to some extent, with no missing values so there was not too much data cleaning required, hence our focus will be on Exploratory Data Analysis in the majority of the assignment.

### *Bank Data*

The input variables from the dataset include

1. Age(numeric).
2. Job (categorical) includes different types of jobs such as 'admin', 'services', 'student', 'technician', 'unemployed' 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'unknown'.
3. Marital (categorical) includes marital status such as 'divorced', 'married', 'single', 'unknown'.
4. Education (categorical) includes education status such as 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown'.
5. Default (categorical) indicates if the client has defaulted or not, with options 'yes', 'no', 'unknown'.
6. Housing (categorical) reflects if the client has a housing loan on his profile, with options 'no', 'yes', 'unknown'.
7. Loan (categorical) shows if the client has obtained a personal loan, with options 'no', 'yes', 'unknown'.
8. Contact (categorical) informs is about contact communication type, with options 'cellular', 'telephone'.
9. Month (categorical) indicates the last contact month of year with the client, including 'jan', 'feb', 'mar', 'april', 'may', 'june', 'july', 'aug', 'sept', 'oct', 'nov', 'dec'.
10. Day of week (categorical) points at the last contact day of the week with the clients: :'mon', 'tue', 'wed', 'thu', 'fri'.
11. Duration (numeric) covers the last contact duration—the measuring unit is in seconds.
12. Campaign (numeric) measures the number of contacts established with the client during the marketing campaign.
13. Pdays (numeric) shows the number of days passed by when the client was last contacted as compared to the previous campaign.

14. Poutcome (categorical) shows the outcome of the marketing campaign from the previous time period, with options 'failure', 'nonexistent', 'success'.
15. Emp.var.rate is an abbreviated term for Employment Variation Rate, and it is a numeric variable with a quarterly indicator;
16. Cons.price.idx means Consumer Price Index, and it is a numeric variable with the monthly indicator.
17. Cons.conf.idx means Consumer Confidence Index, and it is a numeric variable with a monthly indicator.
18. Euribor3m reflects euribor 3 month rate, and it is a numeric variable with a daily indicator;
19. Nr.employedshows the Number of Employees, and it is a numeric variable with a quarterly indicator. The output variable is 'y' that is our desired target, with the option 'yes', and 'no'.

## ***Libraries Used***

There were many libraries used in this assignment.

1. Pandas
2. Numpy
3. Seaborn
4. Matplot
5. Sklearn

## Exploratory Data Analysis

In [205]:

```python
#importing the necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Algorithms
from sklearn.preprocessing import LabelEncoder
from sklearn import linear_model
from imblearn.over_sampling import RandomOverSampler, SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import f1_score, make_scorer
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn import tree



#Accuracy
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE

```

In [206]:

```python
bankdata = pd.read_csv(r'C:\Users\hassa\Desktop\studyhard\bank-additiona
```

In [207]:

```python
#Pandas doesn't allow to see columns at once so dividing in chunks of 10
bankdata[list(bankdata.columns)[:10]].head()
```

Out[207]:

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

In [209]:
```
print(bankdata.shape)
```
```
(41188, 21)
```

In [210]:
```
print(bankdata.size)
```
```
864948
```

In [211]:
```
bankdata.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

In [212]:
```
bankdata.describe().T
```

Out[212]:

| | count | mean | std | min | 25% | 50% | 75% | |
|---|---|---|---|---|---|---|---|---|
| **age** | 41188.0 | 40.024060 | 10.421250 | 17.000 | 32.000 | 38.000 | 47.000 | 98 |
| **duration** | 41188.0 | 258.285010 | 259.279249 | 0.000 | 102.000 | 180.000 | 319.000 | 4918 |
| **campaign** | 41188.0 | 2.567593 | 2.770014 | 1.000 | 1.000 | 2.000 | 3.000 | 56 |
| **pdays** | 41188.0 | 962.475454 | 186.910907 | 0.000 | 999.000 | 999.000 | 999.000 | 999 |
| **previous** | 41188.0 | 0.172963 | 0.494901 | 0.000 | 0.000 | 0.000 | 0.000 | 7 |
| **emp.var.rate** | 41188.0 | 0.081886 | 1.570960 | -3.400 | -1.800 | 1.100 | 1.400 | 1 |
| **cons.price.idx** | 41188.0 | 93.575664 | 0.578840 | 92.201 | 93.075 | 93.749 | 93.994 | 94 |
| **cons.conf.idx** | 41188.0 | -40.502600 | 4.628198 | -50.800 | -42.700 | -41.800 | -36.400 | -26 |
| **euribor3m** | 41188.0 | 3.621291 | 1.734447 | 0.634 | 1.344 | 4.857 | 4.961 | 5 |
| **nr.employed** | 41188.0 | 5167.035911 | 72.251528 | 4963.600 | 5099.100 | 5191.000 | 5228.100 | 5228 |

## Unique and Missing Values Analysis

In [213]:
```
bankdata.nunique()
```

Out[213]:
```
age                 78
job                 12
marital              4
education            8
default              3
housing              3
loan                 3
contact              2
month               10
day_of_week          5
duration          1544
campaign            42
pdays               27
previous             8
poutcome             3
emp.var.rate        10
cons.price.idx      26
cons.conf.idx       26
euribor3m          316
nr.employed         11
y                    2
dtype: int64
```

In [214]:

```
# function to get all unique values in the categorical variables
def unique_val(bankdata):
    bankdata_columns = bankdata.columns
    for i in bankdata_columns:
        if bankdata[i].dtype == 'O':
            print('Unique values in',i,'are',bankdata[i].unique())
unique_val(bankdata)
```

```
Unique values in job are ['housemaid' 'services' 'admin.' 'blue-collar' 'te
chnician' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']
Unique values in marital are ['married' 'single' 'divorced' 'unknown']
Unique values in education are ['basic.4y' 'high.school' 'basic.6y' 'basic.
9y' 'professional.course'
 'unknown' 'university.degree' 'illiterate']
Unique values in default are ['no' 'unknown' 'yes']
Unique values in housing are ['no' 'yes' 'unknown']
Unique values in loan are ['no' 'yes' 'unknown']
Unique values in contact are ['telephone' 'cellular']
Unique values in month are ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar'
'apr' 'sep']
Unique values in day_of_week are ['mon' 'tue' 'wed' 'thu' 'fri']
Unique values in poutcome are ['nonexistent' 'failure' 'success']
Unique values in y are ['no' 'yes']
```

In [215]:

```
print('Sum of missing values')
bankdata.isnull().sum()
```

```
Sum of missing values
```

Out[215]:

```
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 [216]:
```
#count the number of rows for each type
bankdata.groupby('y').size()
```

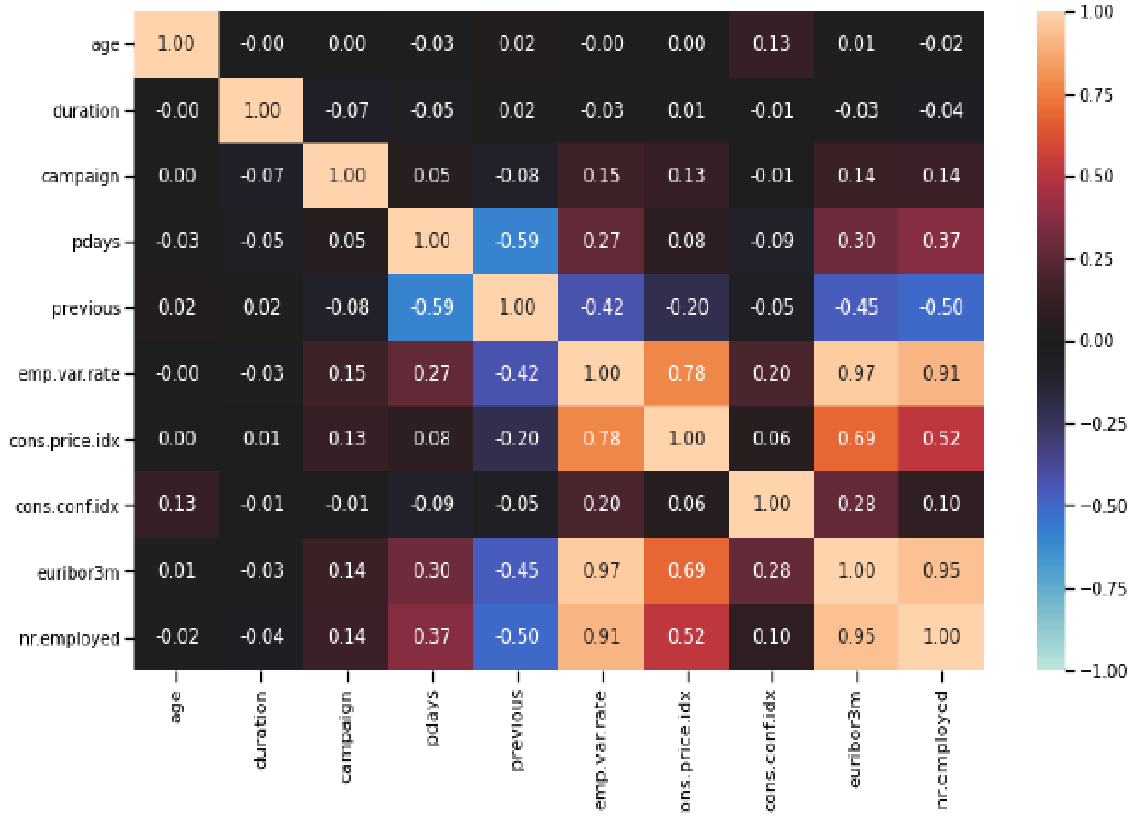
Out[216]:
```
y
no     36548
yes     4640
dtype: int64
```

In [217]:
```
# Percentage of Yes and No
print("\nPercentage of value count in y\n",
      bankdata.y.value_counts(normalize=True)*100)
```

```
Percentage of value count in y
 no     88.734583
yes    11.265417
Name: y, dtype: float64
```

In [218]:

```
# using heatmap to visualize correlation between the columns
plt.figure(figsize = (13,7))
ax = sns.heatmap(bankdata.corr(), annot=True, fmt='.2f',
                 vmin=-1, vmax=1, center= 0)
```



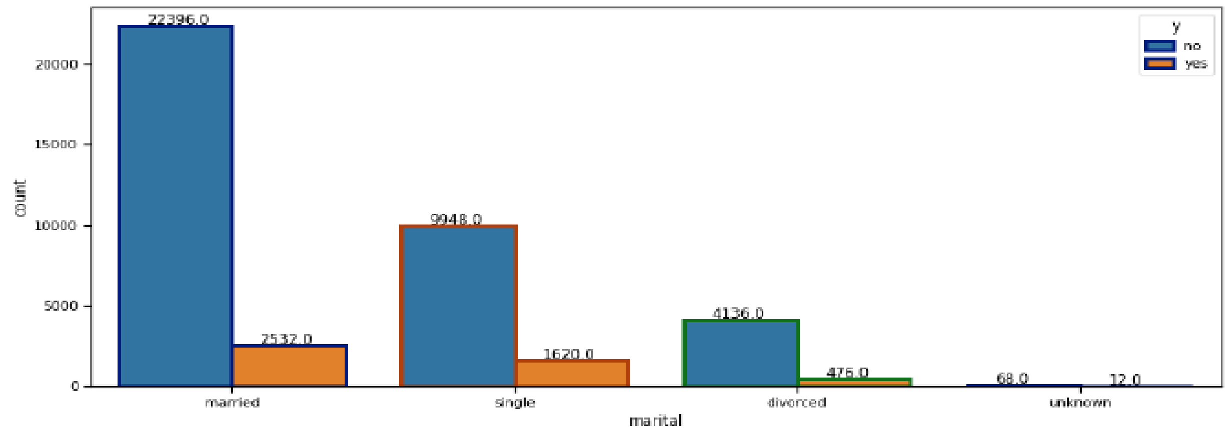

As shown in the above picture, Here we found out the correlations of the features available in the bank data. The correlation was all of numerical features.

1. From the output of correlation matrix , we can see that : age, duration, campaign, pdays, previous, emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m and nr.employed are numerical variables.
2. pdays and previous are negatively correlated (-0.59)
3. euribor3m and nr.employed are positively correlated .95
4. euribor3m and emp.var.rate are positively correlated .97
5. Majority of the data set variables don't have any correlation with each other.

## Bivariate Analysis on Categorical Variables

In [219]:

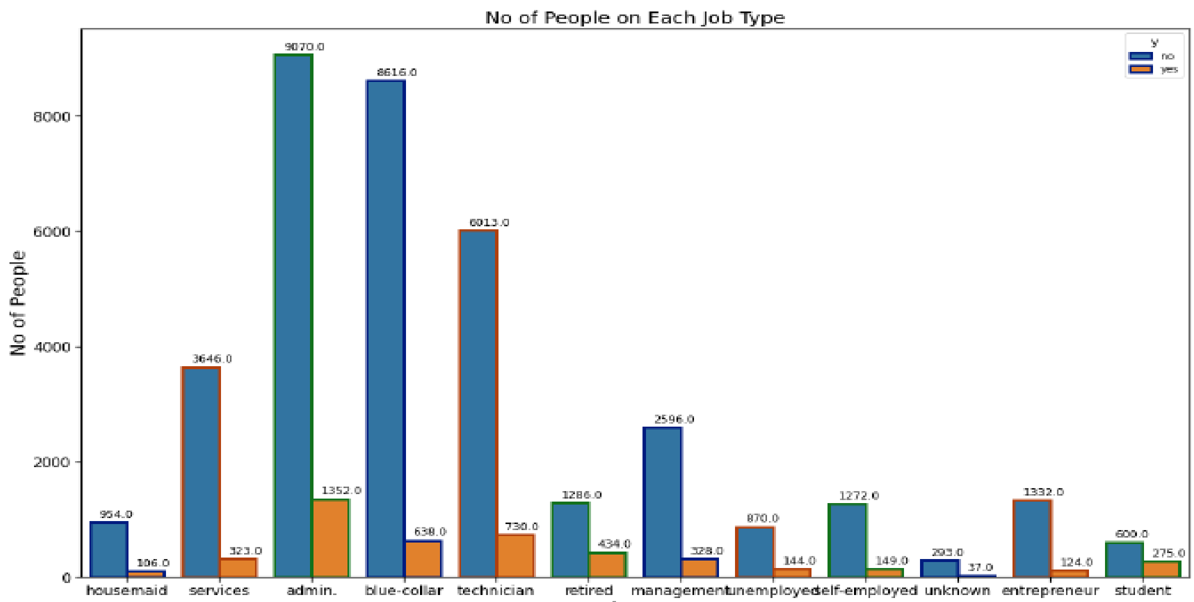
```
fig, ax = plt.subplots()
fig.set_size_inches(16, 6)
ax = sns.countplot(x='marital',hue='y',data=bankdata,
                   linewidth=3,
                   edgecolor=sns.color_palette("dark", 3))

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.ge
```




In the above marital visualization we can see that marital status does have a very high effect if a person wants to sign up for a term deposit or not. Since signing up for a term deposit is a very serious financial decision thus marital status needs to be considered while we research on the dataset.

In [220]:

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 12)
sns.countplot(x = 'job', hue='y', data = bankdata,linewidth=3, edgecolor=
ax.set_xlabel('Job Type', fontsize=20)
ax.set_ylabel('No of People', fontsize=20)
ax.set_title('No of People on Each Job Type', fontsize=20)
ax.tick_params(labelsize=15)

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.ge
```

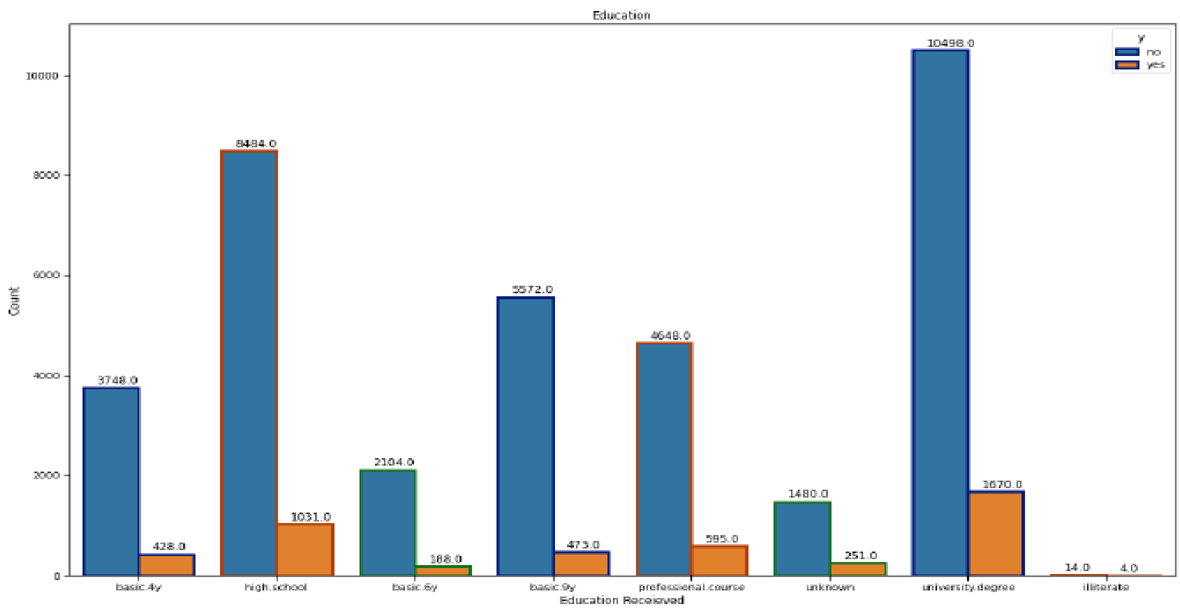



In the  Figure above,a person's financial strength can be seen by the type of job he or she is doing. Also a person's knowledge towards investment can also be seen by the type of job he is into. The  job type  admin, blue collar and technicians  in subscription of the term deposit are highest

In [221]:

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 12)
ax = sns.countplot(x = 'education',hue='y',data=bankdata, linewidth=3, ec
 
for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.g
 
ax.set_xlabel('Education Receieved', fontsize=12)
ax.set_ylabel('Count', fontsize=12)
ax.set_title('Education', fontsize=12)
```

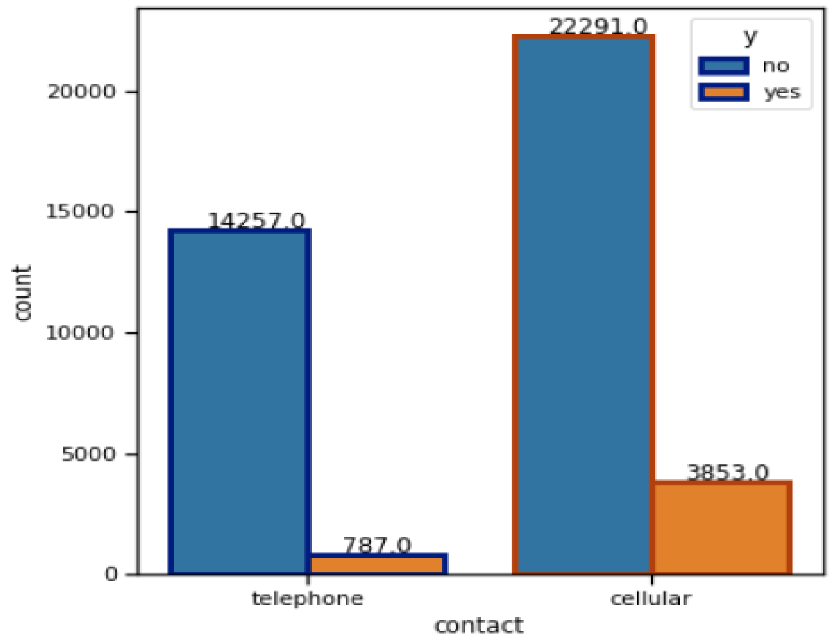
Out[221]: Text(0.5, 1.0, 'Education')




In the above visualization we see how term deposits can be affected by the level of education. Education level brings financial knowledge towards various types of investments so it can play an important role in subscription of the trade deposit.

In [222]:

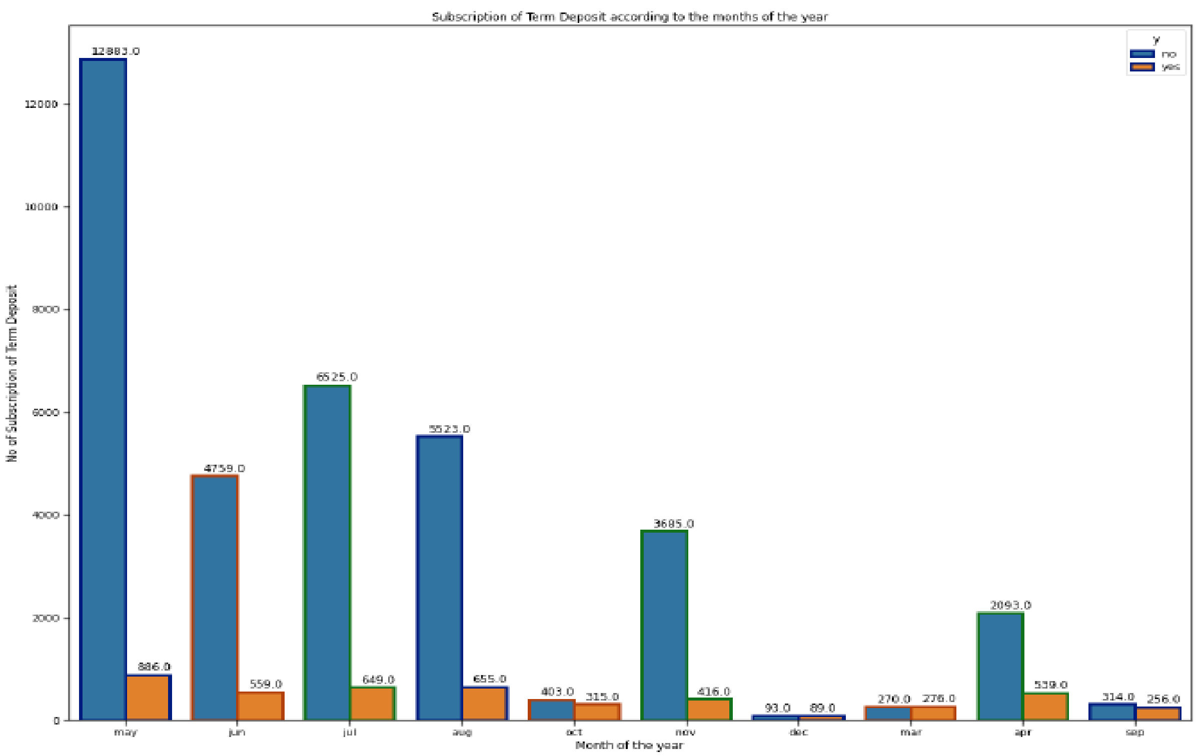
```
fig, ax = plt.subplots()
fig.set_size_inches(6, 6)

sns.countplot(x='contact',hue='y',data=bankdata, linewidth=3, edgecolor=
 
for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.ge
```




In the above visualization we can see that people using cell phone will sign up more (14 % )as compare to landline (5%) so best practise is to call cellphone for max term deposit outcome

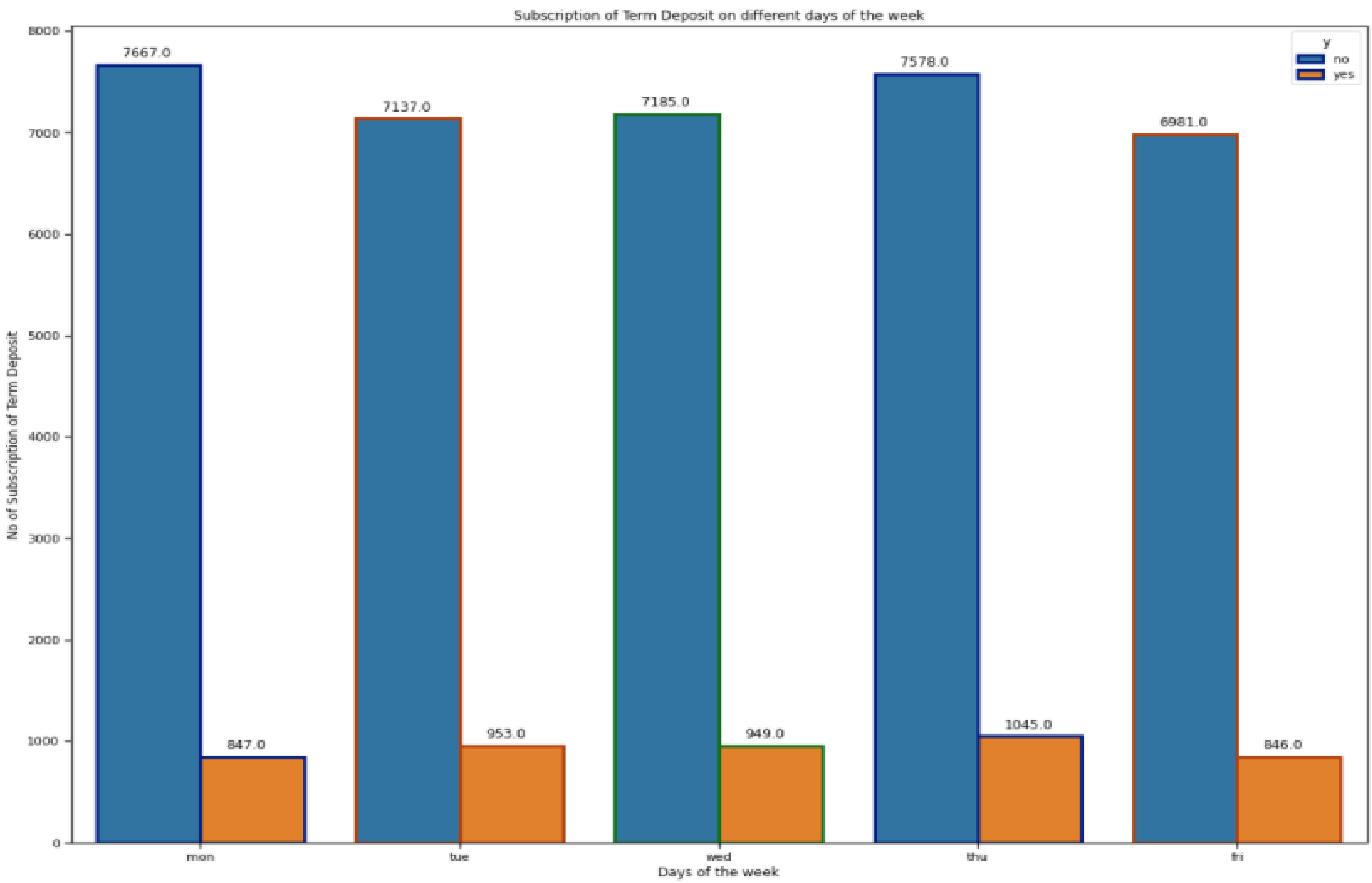
In [223]:

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 15)

sns.countplot(x='month',hue='y',data=bankdata, linewidth=3, edgecolor=sns

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.ge

        plt.title("Subscription of Term Deposit according to the months o
        plt.ylabel("No of Subscription of Term Deposit")
        plt.xlabel("Month of the year")
```



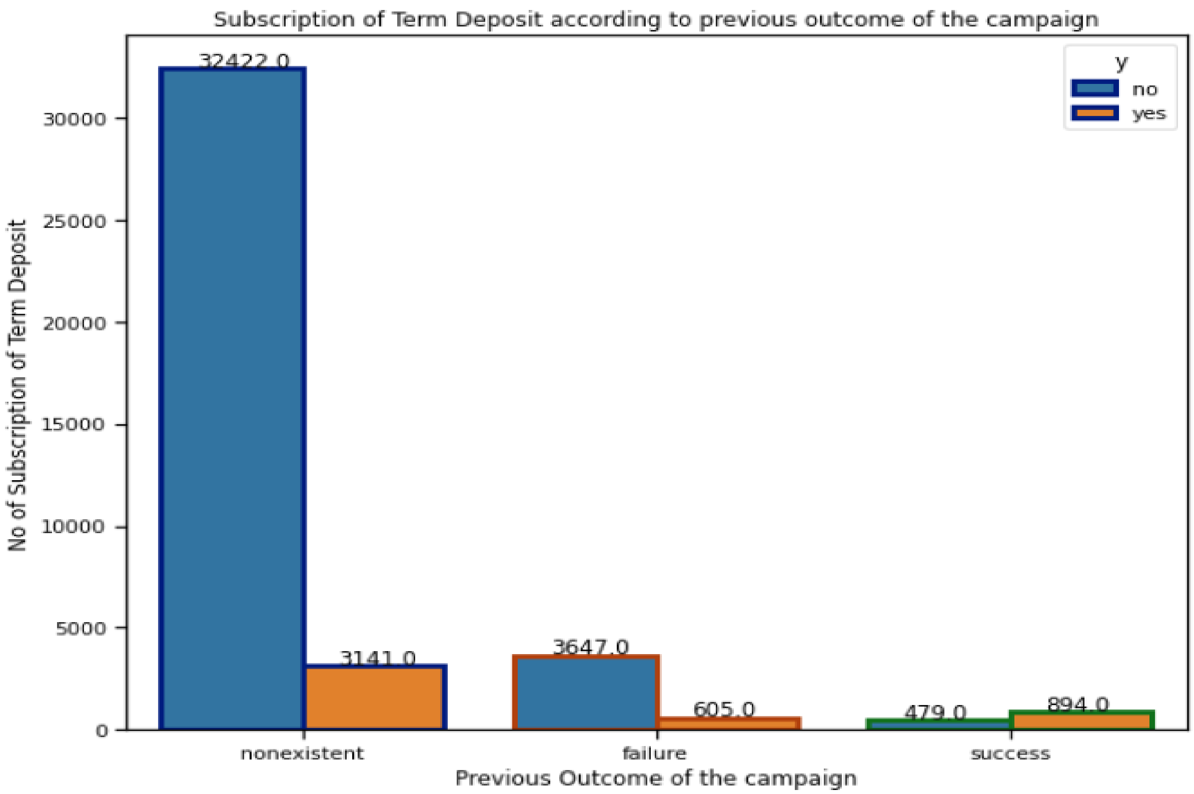

The above visualization is important because a person's financial planning changes according to the months of the year. People have more money in different months of the year. A right time for term deposit campaign is important. We visualized this graph to see how each month is doing with respect to the subscription of the term deposit.

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 15)

ax = sns.countplot(x='day_of_week',hue='y',data=bankdata, linewidth=3, edgecolor=sns.color_palette("dark", 3))

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+80))

        plt.title("Subscription of Term Deposit on different days of the week")
        plt.ylabel("No of Subscription of Term Deposit")
        plt.xlabel("Days of the week")
```



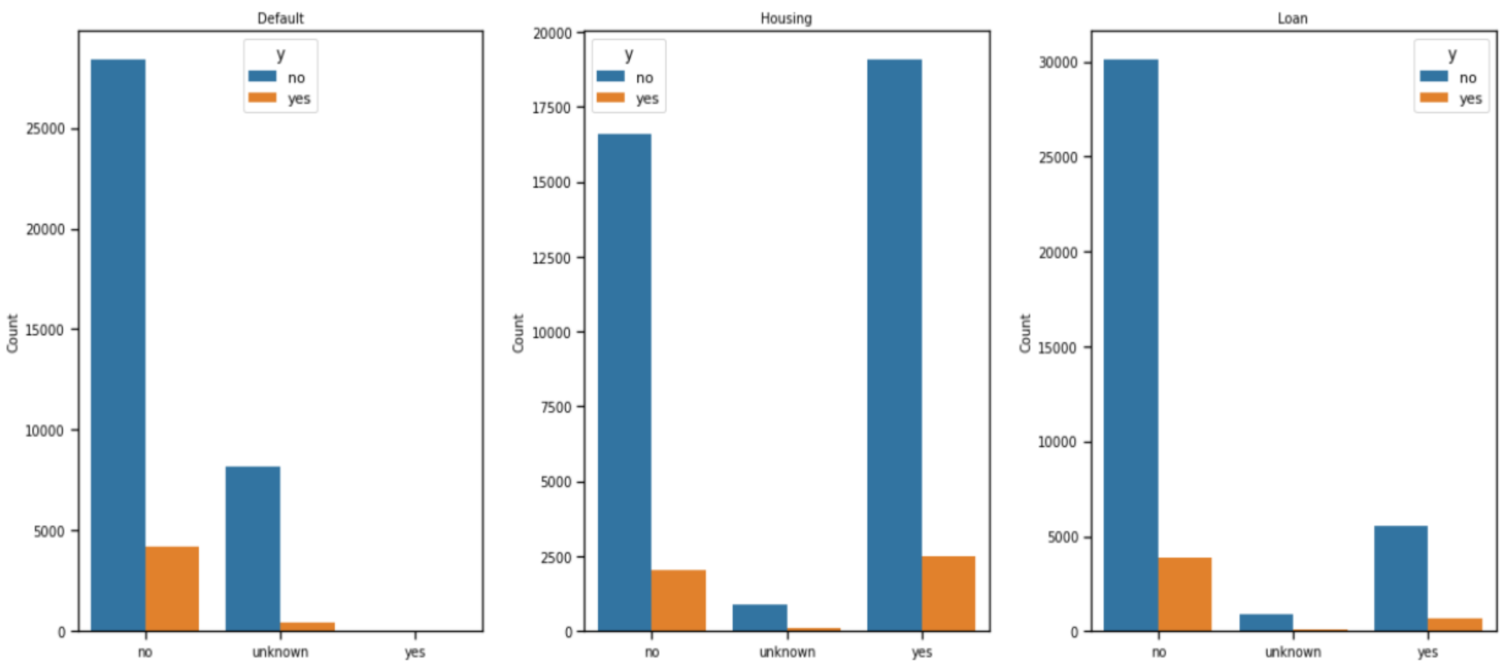

Viz above show's, day of the week also has a huge impact on term deposit subscription .Monday and Thursday most likely has more favourable outcome based on client data

```
In [225]:
fig, ax = plt.subplots()
fig.set_size_inches(10, 8)

ax = sns.countplot(x='poutcome',hue='y',data=bankdata, linewidth=3, edge

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.ge

        plt.title("Subscription of Term Deposit according to previous out
        plt.ylabel("No of Subscription of Term Deposit")
        plt.xlabel("Previous Outcome of the campaign")
```




How likely is that a person will sign up for a similar term deposit next time? This graph is important for the bank so the bank can plan its campaign for customers that have previous term deposits.

```python
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (20,8))
sns.countplot(x = 'default',hue='y', data = bankdata , ax = ax1, order = ['no', 'unknown', 'yes'])
ax1.set_title('Default', fontsize=10)
ax1.set_xlabel('')
ax1.set_ylabel('Count', fontsize=10)
ax1.tick_params(labelsize=10)

sns.countplot(x = 'housing', hue='y',data = bankdata, ax = ax2, order = ['no', 'unknown', 'yes'])
ax2.set_title('Housing', fontsize=10)
ax2.set_xlabel('')
ax2.set_ylabel('Count', fontsize=10)
ax2.tick_params(labelsize=10)

sns.countplot(x = 'loan',hue='y', data = bankdata, ax = ax3, order = ['no', 'unknown', 'yes'])
ax3.set_title('Loan', fontsize=10)
ax3.set_xlabel('')
ax3.set_ylabel('Count', fontsize=10)
ax3.tick_params(labelsize=10)

plt.subplots_adjust(wspace=0.25)
```




Default, Housing and Loan gives an insight from the financial standpoint of a customer. This visualization can help us see what our dataset looks like and what we can predict from this dataset.

## *Separating the Data Set*

We decided to separate the data into 3 parts:

Client Data : 1-7 columns/variables
Marketing Data: 8-15 columns/variables
Economic Data: Bucket with remaining features

We've divided the data with bins of specific attributes/columns. We have leveraged univariate analysis to check the spread of each categorical variable within each bin. During this process, we ran commands to check outliers and fixed the problems using boxplot, imputation using one-hot encoding techniques. As a result of this exercise, we got a thorough understanding of our data, derived more insights and meaningful information for further analysis.

## *1.Client Data*

In [227]:

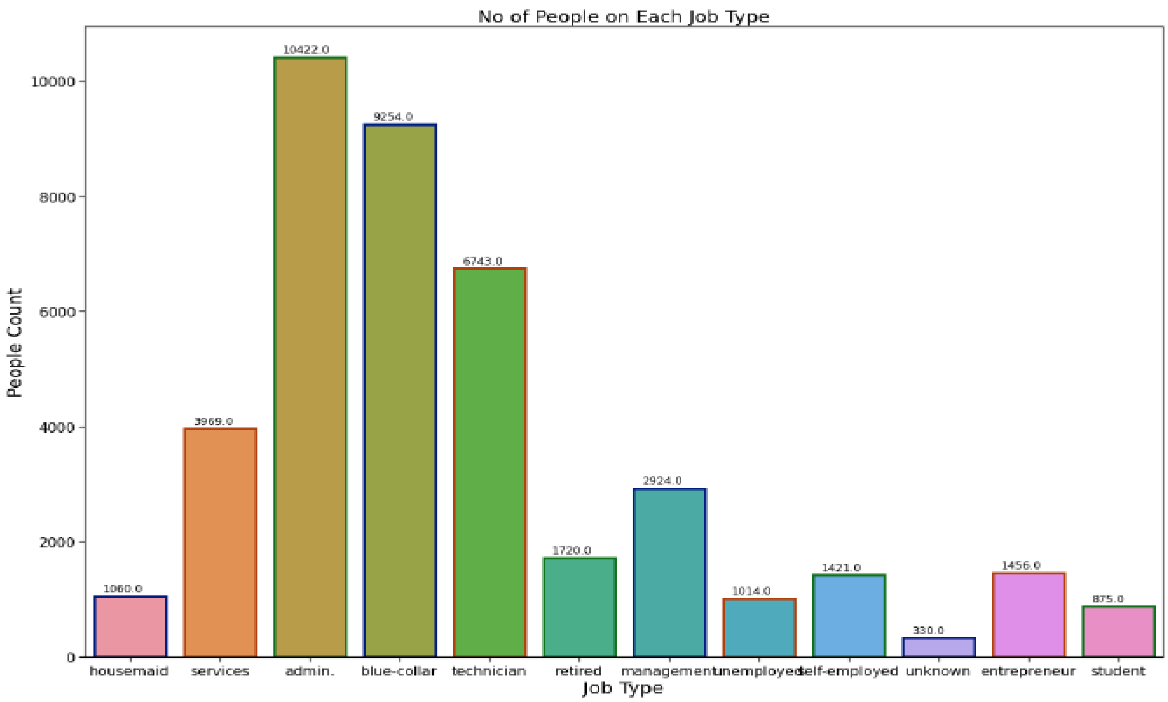
```
#We've divided the data with bins of first  7 columns
clientbankdata = bankdata[["age", "job","marital", "education","default"
clientbankdata.head()
```

Out[227]:

| | age | job | marital | education | default | housing | loan |
|---|---|---|---|---|---|---|---|
| **0** | 56 | housemaid | married | basic.4y | no | no | no |
| **1** | 57 | services | married | high.school | unknown | no | no |
| **2** | 37 | services | married | high.school | no | yes | no |
| **3** | 40 | admin. | married | basic.6y | no | no | no |
| **4** | 56 | services | married | high.school | no | no | yes |

In [228]:

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 14)
sns.countplot(x = 'job',data = clientbankdata,linewidth=3, edgecolor=sns
ax.set_xlabel('Job Type', fontsize=20)
ax.set_ylabel(' People Count ', fontsize=20)
ax.set_title('No of People on Each Job Type', fontsize=20)
ax.tick_params(labelsize=15)

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.ge
```



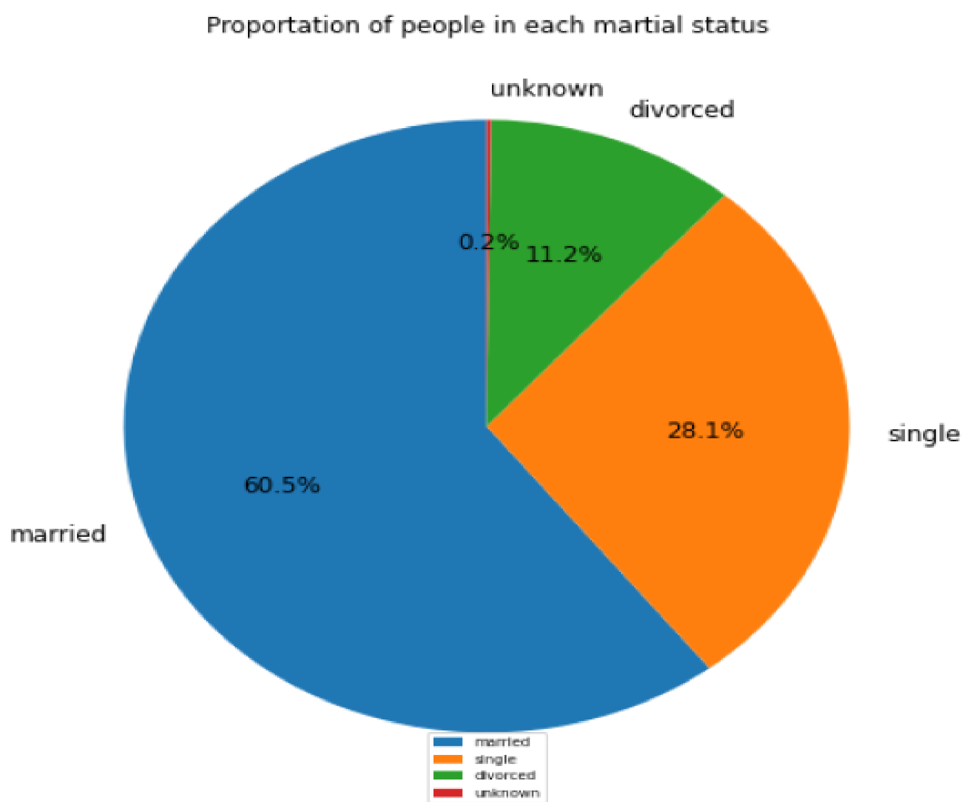

This Univariant graph shows the distribution of people in our dataset according to the job type. This gives an overview of what sample dataset we are dealing with.

In [229]:
```
clientbankdata['marital'].value_counts()
```

Out[229]:
```
married     24928
single      11568
divorced     4612
unknown        80
Name: marital, dtype: int64
```

In [230]:
```
df4 = pd.DataFrame({"Marital Status":["married", "single" , "divorced" ,
_, ax = plt.subplots(figsize = (8,8))
wedges,_,_ = ax.pie(df4['sum']
                    ,labels=df4["Marital Status"]
                    ,shadow=False,startangle=90, autopct="%1.1f%%"
                    ,textprops={'fontsize': 12})
ax.legend(wedges,df4["Marital Status"], loc="lower center", prop={'size'
plt.title('Proportion of people in each martial status')

```

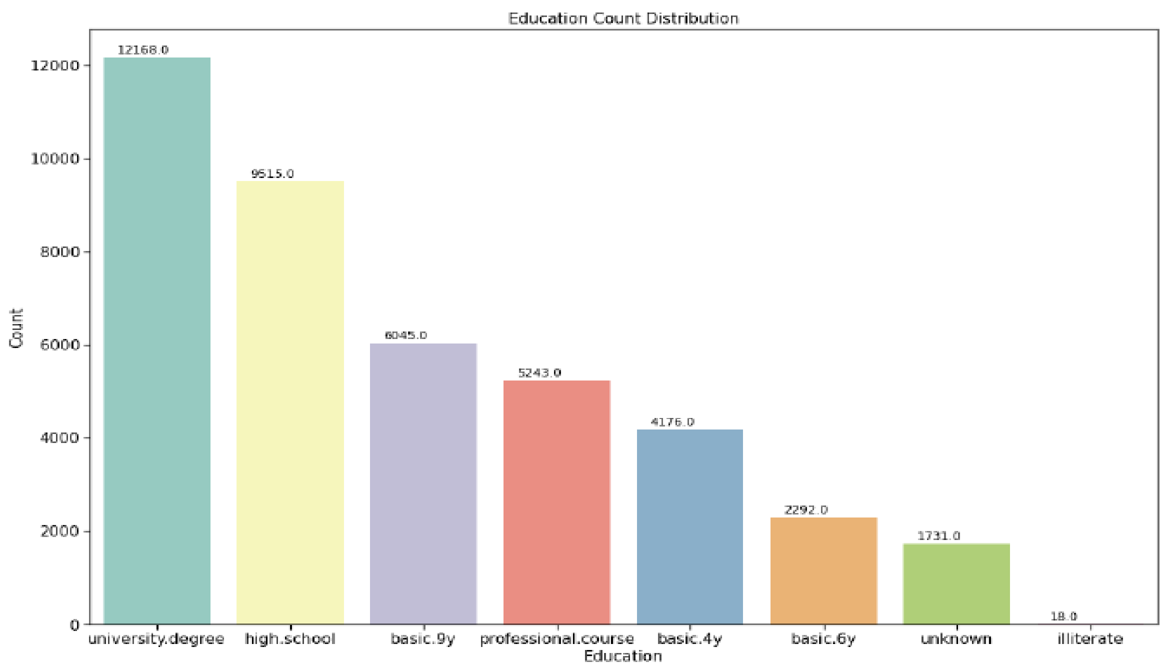
Out[230]: `Text(0.5, 1.0, 'Proportion of people in each martial status')`




In the above graph it shows most of the bank clients are married couple and their is a high probability that they will sign up as a result shared in univariate analysis above.

In [231]:

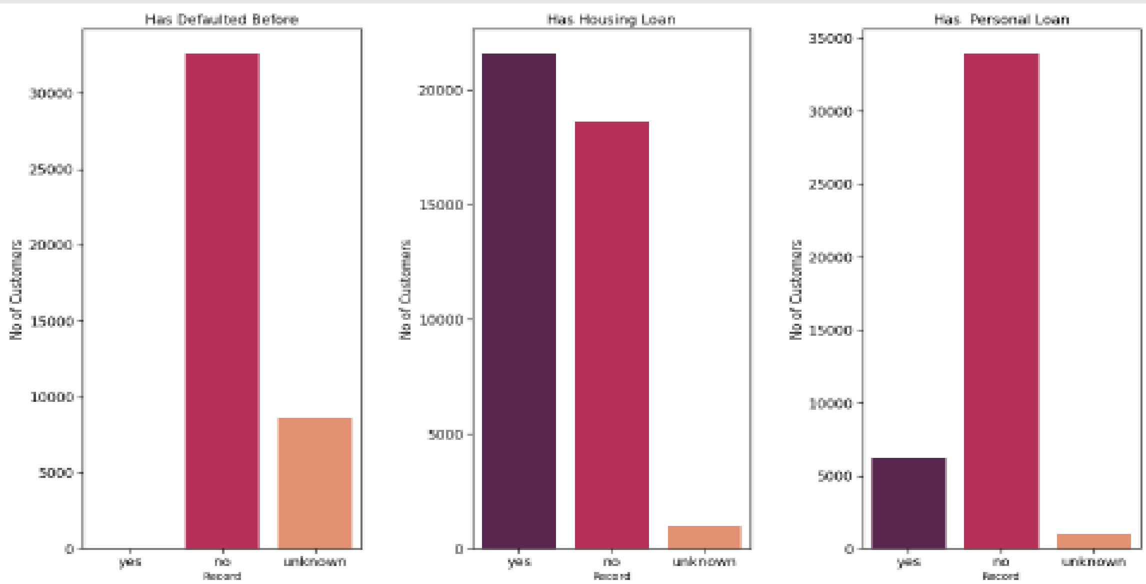
```
fig, ax = plt.subplots()
fig.set_size_inches(18, 12)
sns.countplot(x = 'education', data = clientbankdata,palette="Set3", ord

ax.set_xlabel('Education', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Education Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.ge
```




The Figure above shows a univariate graph of Education Level in ascending order with University degree holders showing the highest number among all education level.

In [233]:

```
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (20,

sns.countplot(x = 'default', data = clientbankdata, ax = ax1, order = ['y
ax1.set_title('Has Defaulted Before', fontsize=15)
ax1.set_xlabel('Record')
ax1.set_ylabel('No of Customers', fontsize=15)
ax1.tick_params(labelsize=15)

sns.countplot(x = 'housing', data = clientbankdata, ax = ax2, order = ['y
ax2.set_title('Has Housing Loan', fontsize=15)
ax2.set_xlabel('Record')
ax2.set_ylabel('No of Customers', fontsize=15)
ax2.tick_params(labelsize=15)

sns.countplot(x = 'loan', data = clientbankdata, ax = ax3, order = ['yes
ax3.set_title('Has  Personal Loan', fontsize=15)
ax3.set_xlabel('Record')
ax3.set_ylabel('No of Customers', fontsize=15)
ax3.tick_params(labelsize=15)

plt.subplots_adjust(wspace=0.4)

```

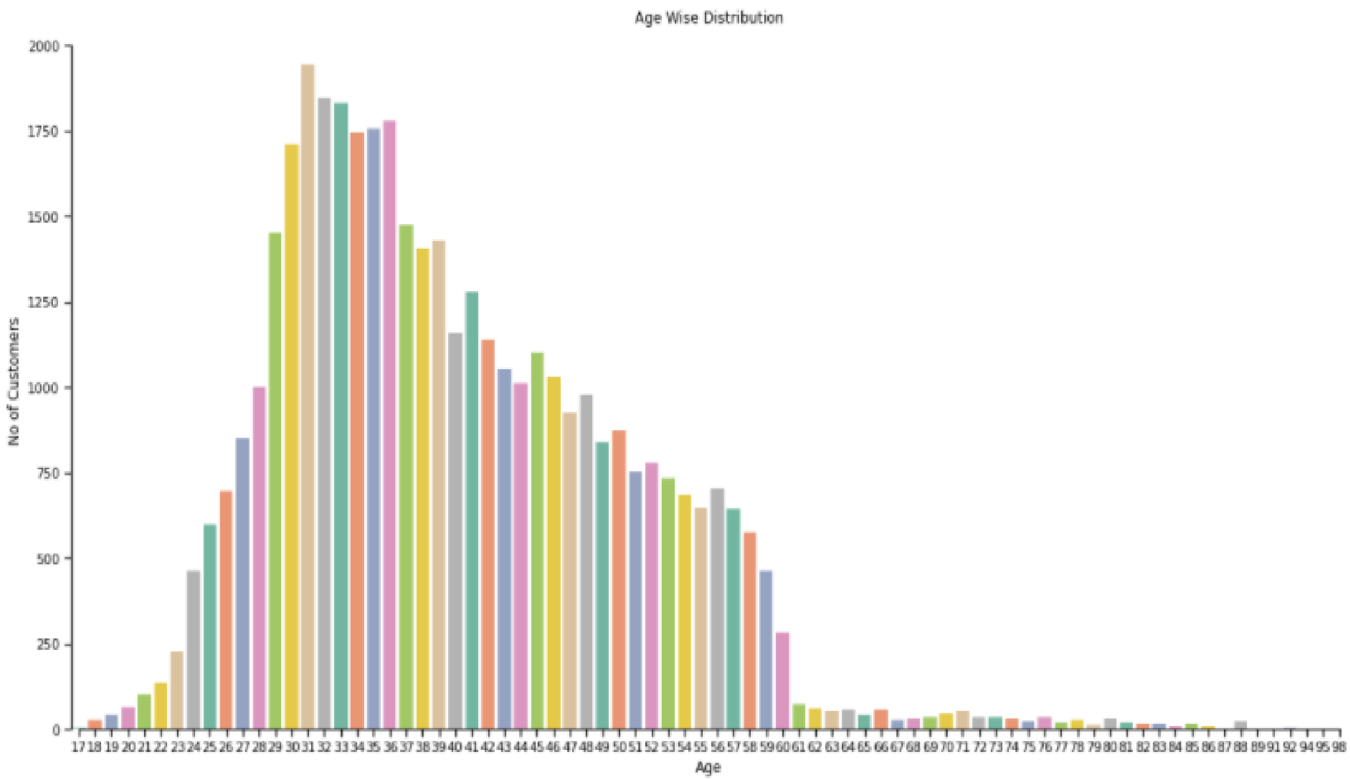



Default, Housing and Loan gives an insight of the financial standing of a customer. This Visualization can help us see how our dataset looks like and what can we predict from this dataset

# Imputation and Outliers Study

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 10)
sns.countplot(x = 'age', data = clientbankdata,palette="Set2")
ax.set_xlabel('Age', fontsize=12)
ax.set_ylabel('No of Customers', fontsize=12)
ax.set_title('Age Wise Distribution', fontsize=12)
sns.despine(trim=True)
```

*Age Feature*

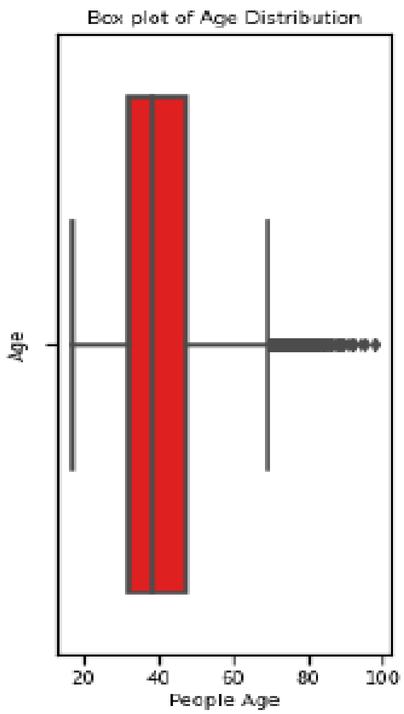


***Age has outliers so using boxplot to find them***

In [235]:

```
fig, (ax) = plt.subplots(nrows = 1, ncols = 1, figsize = (3,7))
sns.boxplot(x = 'age', data = clientbankdata,orient = 'v',color="red")
ax.set_xlabel('People Age', fontsize=10)
ax.set_ylabel('Age', fontsize=10)
ax.set_title('Box plot of Age Distribution', fontsize=10)
ax.tick_params(labelsize=10)
```

```
C:\Users\hassa\anaconda3\lib\site-packages\seaborn\_core.py:1319: UserWarni
ng: Vertical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("Vertical", "x"))
```



In [236]:
```
print(clientbankdata.select_dtypes(include='number').isnull().sum())
```

```
age    0
dtype: int64
```

In [237]:
```
#Create Copy of  clientbankdata for outliers impulation using mean
clientbankdata_copy1 = clientbankdata.copy(deep=True)
clientbankdata_copy1.head()
```

Out[237]:

| | age | job | marital | education | default | housing | loan |
|---|---|---|---|---|---|---|---|
| 0 | 56 | housemaid | married | basic.4y | no | no | no |
| 1 | 57 | services | married | high.school | unknown | no | no |
| 2 | 37 | services | married | high.school | no | yes | no |
| 3 | 40 | admin. | married | basic.6y | no | no | no |
| 4 | 56 | services | married | high.school | no | no | yes |

In [238]:
```
Q1=clientbankdata_copy1['age'].quantile(q = 0.25)
Q2=clientbankdata_copy1['age'].quantile(q = 0.50)
Q3=clientbankdata_copy1['age'].quantile(q = 0.75)
Q4=clientbankdata_copy1['age'].quantile(q = 1.00)
IQR= Q3-Q1

print('1st Quartile: ', Q1)
print('2nd Quartile: ', Q2)
print('3rd Quartile: ', Q3)
print('4th Quartile: ', Q4)
print('IQR: ',IQR)

age_below  = Q1-(1.5*IQR)
print('age_below => ' + str(age_below))

age_above = Q3+(1.5*IQR)
print('age_above => ' + str(age_above))
print("\n")


print('Any age below', Q1 - 1.5*(IQR), 'or above', Q3 + 1.5*(IQR), 'can 

```

```
1st Quartile:   32.0
2nd Quartile:   38.0
3rd Quartile:   47.0
4th Quartile:   98.0
IQR:  15.0
age_below => 9.5
age_above => 69.5


Any age below 9.5 or above 69.5 can can be considered as Outliers.
```

In [240]:
```
#replacing outliers with nan
clientbankdata_copy1['age'][((clientbankdata_copy1['age'] < age_below) |
```

In [241]:
```
#finding null values first in age column for imputation purpose
print(clientbankdata_copy1['age'].isnull().sum())
```

```
469
```

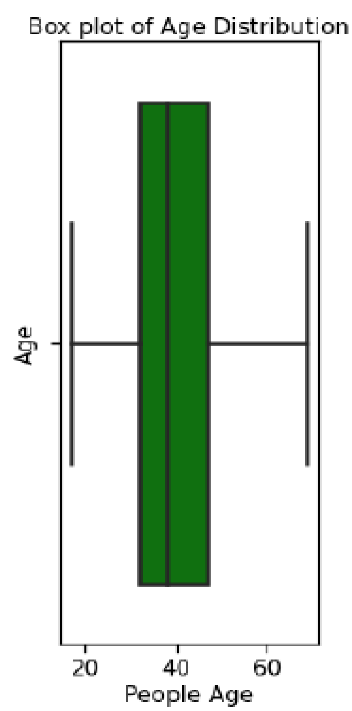
In [242]:
```
column_means = clientbankdata_copy1['age'].mean()
print(column_means)
clientbankdata_copy1['age'] = clientbankdata_copy1['age'].fillna(column_m
```

```
39.599007834180604
```

In [243]:
```
#Imputation with means removed null values
print(clientbankdata_copy1['age'].isnull().sum())
```

***Using Box plot to check if outliers have been removed***

In [244]:
```
fig, (ax) = plt.subplots(nrows = 1, ncols = 1, figsize = (3,8))
sns.boxplot(x = 'age', data = clientbankdata_copy1,orient = 'v',color="gr
ax.set_xlabel('People Age', fontsize=15)
ax.set_ylabel('Age', fontsize=15)
ax.set_title('Box plot of Age Distribution', fontsize=15)
ax.tick_params(labelsize=15)
```




From the above boxplot,it shows  the median age lies between age 38 to 40 for the client who sign up or doesn't sign up for term deposit

***Now We are binning the age into different categories.***

In [245]:
```python
# functions to create binning in age

def age(dframe):
    dframe.loc[dframe['age'] <= 32, 'age'] = 1
    dframe.loc[(dframe['age'] > 32) & (dframe['age'] <= 38), 'age'] = 2
    dframe.loc[(dframe['age'] > 38) & (dframe['age'] <=47), 'age'] = 3
    dframe.loc[(dframe['age'] > 47) & (dframe['age'] <=69), 'age'] = 4
    dframe.loc[(dframe['age'] > 69), 'age']= 5

    return dframe

age(clientbankdata_copy1);
```

In [246]:
```python
clientbankdata_copy1['age'].head()
```

Out[246]:
```
0    4.0
1    4.0
2    2.0
3    3.0
4    4.0
Name: age, dtype: float64
```

In [247]:
```python
# converting age dtype to int
clientbankdata_copy1['age'] = clientbankdata_copy1['age'].astype(int)
```

We've used hot encoding method as shown below to assign 1 and 0 values to new features such as job_admin, job_housemaid,job_management,job_retired etc

In [248]:
```python
clientbankdata_copy1 =pd.get_dummies(clientbankdata_copy1)
clientbankdata_copy1.head()
```

Out[248]:

| | age | job_admin. | job_blue-collar | job_entrepreneur | job_housemaid | job_management | job_retired |
|---|---|---|---|---|---|---|---|
| 0 | 4 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 |
| 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 34 columns

```
In [249]:   1 clientbankdata_copy1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 34 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   age                            41188 non-null  int32
 1   job_admin.                     41188 non-null  uint8
 2   job_blue-collar                41188 non-null  uint8
 3   job_entrepreneur               41188 non-null  uint8
 4   job_housemaid                  41188 non-null  uint8
 5   job_management                 41188 non-null  uint8
 6   job_retired                    41188 non-null  uint8
 7   job_self-employed              41188 non-null  uint8
 8   job_services                   41188 non-null  uint8
 9   job_student                    41188 non-null  uint8
 10  job_technician                 41188 non-null  uint8
 11  job_unemployed                 41188 non-null  uint8
 12  job_unknown                    41188 non-null  uint8
 13  marital_divorced               41188 non-null  uint8
 14  marital_married                41188 non-null  uint8
 15  marital_single                 41188 non-null  uint8
 16  marital_unknown                41188 non-null  uint8
 17  education_basic.4y             41188 non-null  uint8
 18  education_basic.6y             41188 non-null  uint8
 19  education_basic.9y             41188 non-null  uint8
 20  education_high.school          41188 non-null  uint8
 21  education_illiterate           41188 non-null  uint8
 22  education_professional.course  41188 non-null  uint8
 23  education_university.degree    41188 non-null  uint8
 24  education_unknown              41188 non-null  uint8
 25  default_no                     41188 non-null  uint8
 26  default_unknown                41188 non-null  uint8
 27  default_yes                    41188 non-null  uint8
 28  housing_no                     41188 non-null  uint8
 29  housing_unknown                41188 non-null  uint8
 30  housing_yes                    41188 non-null  uint8
 31  loan_no                        41188 non-null  uint8
 32  loan_unknown                   41188 non-null  uint8
 33  loan_yes                       41188 non-null  uint8
dtypes: int32(1), uint8(33)
memory usage: 1.5 MB
```

## 2. Marketing Data Analysis

In [250]:
```
# Creating seperate datasets for marketing related data
bank_marketing = bankdata [["contact","month","day_of_week","duration","
bank_marketing.head()
```

Out[250]:

| | contact | month | day_of_week | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|
| 0 | telephone | may | mon | 261 | 1 | 999 | 0 | nonexistent |
| 1 | telephone | may | mon | 149 | 1 | 999 | 0 | nonexistent |
| 2 | telephone | may | mon | 226 | 1 | 999 | 0 | nonexistent |
| 3 | telephone | may | mon | 151 | 1 | 999 | 0 | nonexistent |
| 4 | telephone | may | mon | 307 | 1 | 999 | 0 | nonexistent |

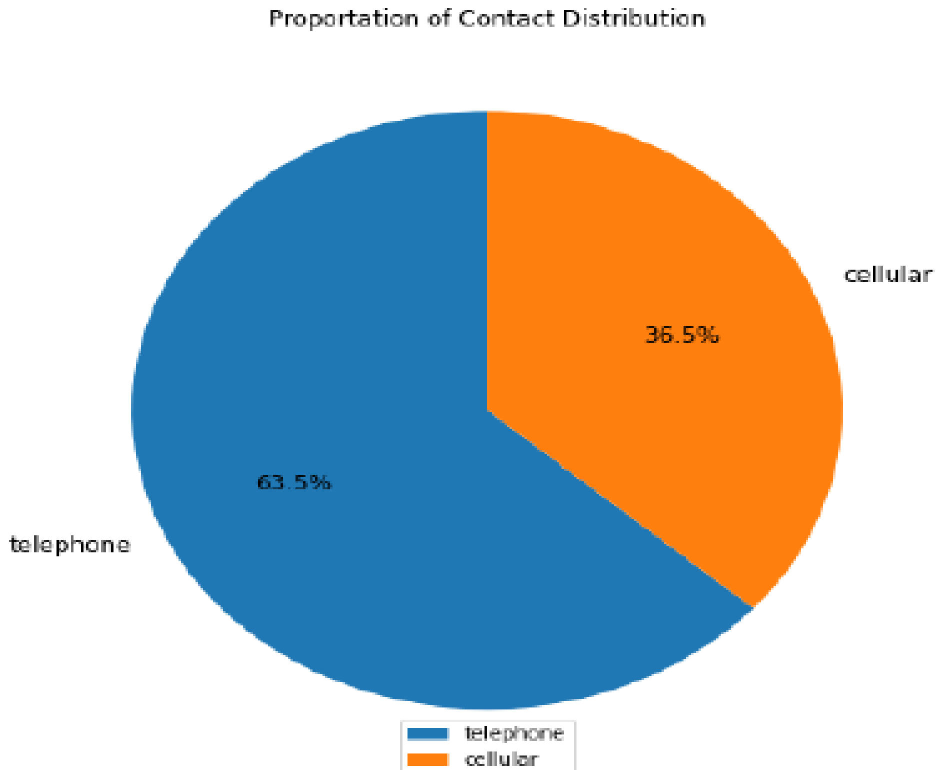
In [251]:
```
bank_marketing['contact'].value_counts()
```

Out[251]:
```
cellular     26144
telephone    15044
Name: contact, dtype: int64
```

In [252]:

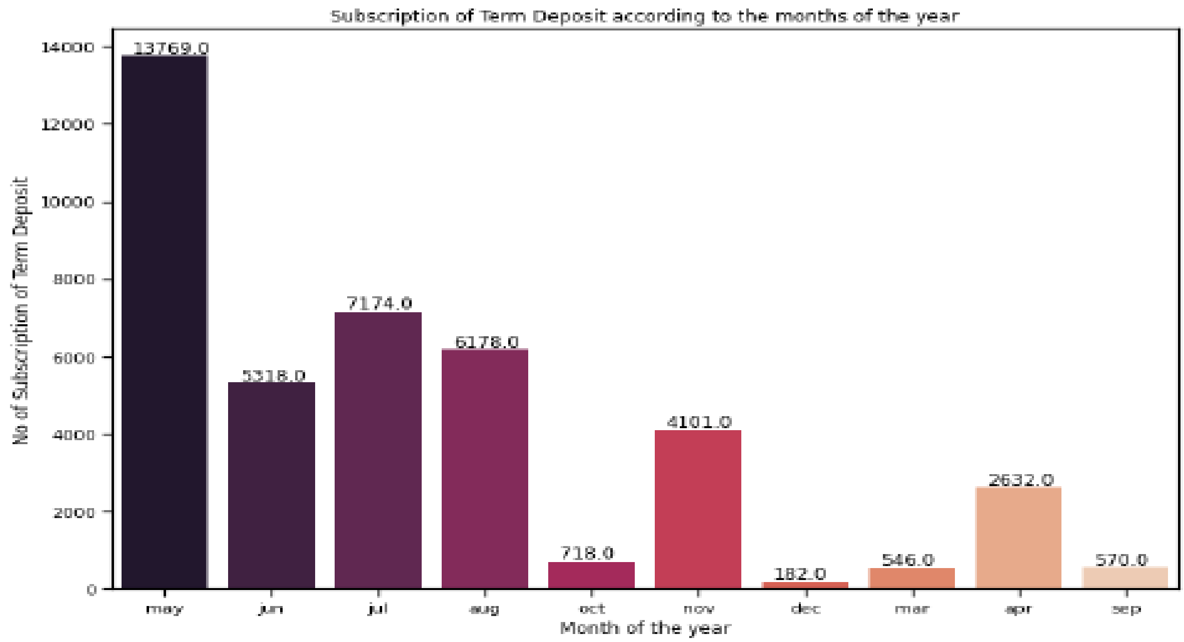
```
df5 = pd.DataFrame({"Contact Mode":["telephone", "cellular"], "sum":[261
_, ax = plt.subplots(figsize = (8,8))
wedges,_,_ = ax.pie(df5['sum']
                    ,labels=df5["Contact Mode"]
                    ,shadow=False,startangle=90, autopct="%1.1f%%"
                    ,textprops={'fontsize': 12})
ax.legend(wedges,df5["Contact Mode"], loc="lower center", prop={'size': 
plt.title('Proportation of Contact Distribution')
```

Out[252]: Text(0.5, 1.0, 'Proportation of Contact Distribution')



In [253]:

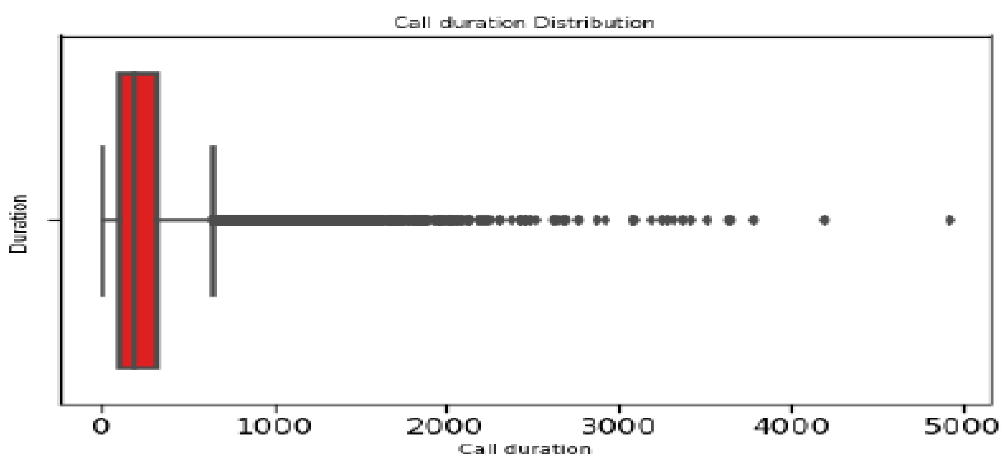
```
plt.figure(figsize = (12,8))
ax = sns.countplot(x='month',data=bank_marketing, linewidth=3, palette="r

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.g

        plt.title("Subscription of Term Deposit according to the months o
        plt.ylabel("No of Subscription of Term Deposit")
        plt.xlabel("Month of the year")
```




In [255]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize = (8,5))
sns.boxplot(x = 'duration', data = bank_marketing, orient = 'v',color="r
ax.set_xlabel('Call duration', fontsize=10)
ax.set_ylabel('Duration', fontsize=10)
ax.set_title('Call duration Distribution', fontsize=10)
ax.tick_params(labelsize=15)
```

```
C:\Users\hassa\anaconda3\lib\site-packages\seaborn\_core.py:1319: UserWarni
ng: Vertical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("vertical", "x"))
```



In [256]:

```
Q1=bank_marketing['duration'].quantile(q = 0.25)
Q2=bank_marketing['duration'].quantile(q = 0.50)
Q3=bank_marketing['duration'].quantile(q = 0.75)
Q4=bank_marketing['duration'].quantile(q = 1.00)

IQR= Q3-Q1


print('1st Quartile: ', Q1)
print('2nd Quartile: ', Q2)
print('3rd Quartile: ', Q3)
print('4th Quartile: ', Q4)
print('IQR: ',IQR)


duration_below  = Q1-(1.5*IQR)
duration_above = Q3+(1.5*IQR)

print('Any duration below', duration_below, 'or above', duration_above,

```

```
1st Quartile:  102.0
2nd Quartile:  180.0
3rd Quartile:  319.0
4th Quartile:  4918.0
IQR:  217.0
Any duration below -223.5 or above 644.5 can can be considered as Outliers.
```

In [257]:

```
#replacing outliers with nan
bank_marketing['duration'][((bank_marketing['duration'] < duration_below
```

In [258]:

```
print(bank_marketing['duration'].isnull().sum())
```

```
2963
```

In [259]:
```
bank_marketing['duration'].sample(30)
```

Out[259]:
```
12873    113.0
12578    353.0
10440    226.0
3617     130.0
38852    365.0
34636    157.0
32211    612.0
1938       NaN
34635     88.0
34551     16.0
8623     103.0
39138    368.0
3078     311.0
31542    187.0
760      117.0
34501      NaN
10665    437.0
524      145.0
28851    263.0
2183     236.0
2039       NaN
18380     87.0
40773    231.0
31399    251.0
40329      NaN
38408     72.0
314      358.0
40315      NaN
25261    255.0
29120    251.0
Name: duration, dtype: float64
```

In [260]:
```
duration_mean = bank_marketing['duration'].mean()
print(duration_mean)
bank_marketing['duration'] = bank_marketing['duration'].fillna(duration_
```

```
203.25483322432962
```

In [261]:
```
print(bank_marketing['duration'].isnull().sum())
```

```
0
```

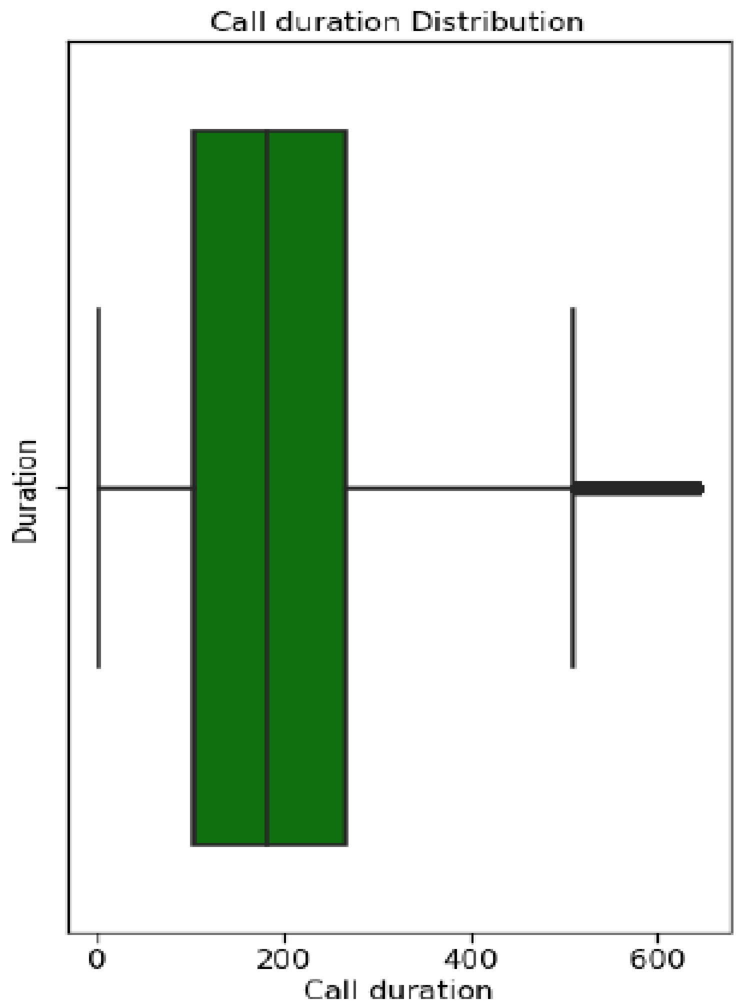
In [262]:
```
# checking other details of age
bank_marketing['duration'].describe()
```

Out[262]:
```
count    41188.000000
mean       203.254833
std        135.850094
min          0.000000
25%        102.000000
50%        180.000000
75%        265.000000
max        644.000000
Name: duration, dtype: float64
```

In [263]:

```
fig, (ax) = plt.subplots(nrows = 1, ncols = 1, figsize = (6,10))
sns.boxplot(x = 'duration', data = bank_marketing,orient = 'v',color="gr
ax.set_xlabel('Call duration', fontsize=15)
ax.set_ylabel('Duration', fontsize=15)
ax.set_title('Call duration Distribution', fontsize=15)
ax.tick_params(labelsize=15)
```

```
C:\Users\hassa\anaconda3\lib\site-packages\seaborn\_core.py:1319: UserWarni
ng: Vertical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("Vertical", "x"))
```




In [264]:

```
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
```

## *Dividing the duration into buckets*

In [265]:

```
#Imputing  duration columns with 1,2,3,4 using bucket method
def duration(df):

    df.loc[df['duration'] <= 102, 'duration'] = 1
    df.loc[(df['duration'] > 102) & (df['duration'] <= 180)  , 'duration
    df.loc[(df['duration'] > 180) & (df['duration'] <= 265)  , 'duration
    df.loc[(df['duration'] > 265) & (df['duration'] <= 644), 'duration']
    df.loc[df['duration']  > 644, 'duration'] = 5
    return df
duration(bank_marketing).head()
```

Out[265]:

| | contact | month | day_of_week | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|
| **0** | telephone | may | mon | 3.0 | 1 | 999 | 0 | nonexistent |
| **1** | telephone | may | mon | 2.0 | 1 | 999 | 0 | nonexistent |
| **2** | telephone | may | mon | 3.0 | 1 | 999 | 0 | nonexistent |
| **3** | telephone | may | mon | 2.0 | 1 | 999 | 0 | nonexistent |
| **4** | telephone | may | mon | 4.0 | 1 | 999 | 0 | nonexistent |

In [266]:

```
d_mons = {'jan':1, 'feb':2, 'mar':3, 'apr':4, 'may':5,
    'jun':6, 'jul':7, 'aug':8, 'sep':9, 'oct':10,
    'nov':11, 'dec':12}

bank_marketing.month=bank_marketing.month.map(d_mons)
```

In [267]:

```
bank_marketing.head()
```

Out[267]:

| | contact | month | day_of_week | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|
| **0** | telephone | 5 | mon | 3.0 | 1 | 999 | 0 | nonexistent |
| **1** | telephone | 5 | mon | 2.0 | 1 | 999 | 0 | nonexistent |
| **2** | telephone | 5 | mon | 3.0 | 1 | 999 | 0 | nonexistent |
| **3** | telephone | 5 | mon | 2.0 | 1 | 999 | 0 | nonexistent |
| **4** | telephone | 5 | mon | 4.0 | 1 | 999 | 0 | nonexistent |

In [268]:

```
week_day = {'mon':1, 'tue':2, 'wed':3, 'thu':4, 'fri':5,
    'sat':6, 'sun':7}

bank_marketing.day_of_week=bank_marketing.day_of_week.map(week_day)
```

In [269]:

```
bank_marketing[["month", "day_of_week"]] = bank_marketing[["month","day_
```

Mapping function as shown above  is used to convert month and week into numerical values

In [270]:
```
bank_marketing.head()
```

Out[270]:

| | contact | month | day_of_week | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|
| 0 | telephone | 5 | 1 | 3.0 | 1 | 999 | 0 | nonexistent |
| 1 | telephone | 5 | 1 | 2.0 | 1 | 999 | 0 | nonexistent |
| 2 | telephone | 5 | 1 | 3.0 | 1 | 999 | 0 | nonexistent |
| 3 | telephone | 5 | 1 | 2.0 | 1 | 999 | 0 | nonexistent |
| 4 | telephone | 5 | 1 | 4.0 | 1 | 999 | 0 | nonexistent |

In [271]:
```
bank_marketing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   contact      41188 non-null  object 
 1   month        41188 non-null  int64  
 2   day_of_week  41188 non-null  int64  
 3   duration     41188 non-null  float64
 4   campaign     41188 non-null  int64  
 5   pdays        41188 non-null  int64  
 6   previous     41188 non-null  int64  
 7   poutcome     41188 non-null  object 
dtypes: float64(1), int64(5), object(2)
memory usage: 2.5+ MB
```

In [272]:
```
bank_marketing.sample(5)
```

Out[272]:

| | contact | month | day_of_week | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|
| 38561 | cellular | 10 | 4 | 4.0 | 2 | 2 | 2 | success |
| 23437 | cellular | 8 | 3 | 1.0 | 5 | 999 | 0 | nonexistent |
| 38740 | cellular | 11 | 3 | 4.0 | 1 | 999 | 1 | failure |
| 18361 | cellular | 7 | 4 | 2.0 | 4 | 999 | 0 | nonexistent |
| 31789 | cellular | 5 | 4 | 4.0 | 1 | 999 | 0 | nonexistent |

#Label Encoder on bank Marketing Data Machine learning algorithm can only read numerical values. It is therefore essential to encode categorical features into numerical values

In [273]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
bank_marketing['contact'] = le.fit_transform(bank_marketing['contact']).
bank_marketing['poutcome'] = le.fit_transform(bank_marketing['poutcome']
```

In [274]:
```
bank_marketing.to_csv('bank_marketing.csv')
```

In [275]:
```
bank_marketing.head()
```

Out[275]:

| | contact | month | day_of_week | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 5 | 1 | 3.0 | 1 | 999 | 0 | 1 |
| 1 | 1 | 5 | 1 | 2.0 | 1 | 999 | 0 | 1 |
| 2 | 1 | 5 | 1 | 3.0 | 1 | 999 | 0 | 1 |
| 3 | 1 | 5 | 1 | 2.0 | 1 | 999 | 0 | 1 |
| 4 | 1 | 5 | 1 | 4.0 | 1 | 999 | 0 | 1 |

In [276]:
```
bank_marketing.describe()
```

Out[276]:

| | contact | month | day_of_week | duration | campaign | pdays | |
|---|---|---|---|---|---|---|---|
| count | 41188.000000 | 41188.000000 | 41188.000000 | 41188.000000 | 41188.000000 | 41188.000000 | 41 |
| mean | 0.365252 | 6.607896 | 2.979581 | 2.495411 | 2.567593 | 962.475454 | |
| std | 0.481507 | 2.040998 | 1.411514 | 1.117039 | 2.770014 | 186.910907 | |
| min | 0.000000 | 3.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | |
| 25% | 0.000000 | 5.000000 | 2.000000 | 1.000000 | 1.000000 | 999.000000 | |
| 50% | 0.000000 | 6.000000 | 3.000000 | 2.000000 | 2.000000 | 999.000000 | |
| 75% | 1.000000 | 8.000000 | 4.000000 | 3.000000 | 3.000000 | 999.000000 | |
| max | 1.000000 | 12.000000 | 5.000000 | 4.000000 | 56.000000 | 999.000000 | |

In [277]:
```
bank_marketing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   contact      41188 non-null  int32  
 1   month        41188 non-null  int64  
 2   day_of_week  41188 non-null  int64  
 3   duration     41188 non-null  float64
 4   campaign     41188 non-null  int64  
 5   pdays        41188 non-null  int64  
 6   previous     41188 non-null  int64  
 7   poutcome     41188 non-null  int32  
dtypes: float64(1), int32(2), int64(5)
memory usage: 2.2 MB
```

In [278]:

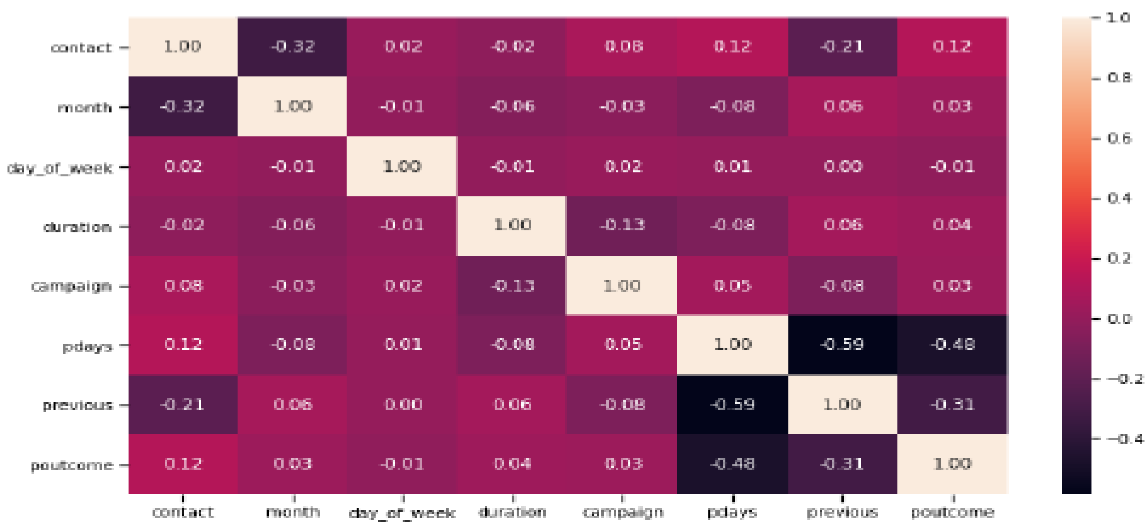
```
corr=bank_marketing.corr()
corr
```

Out[278]:

| | contact | month | day_of_week | duration | campaign | pdays | previous | p |
|---|---|---|---|---|---|---|---|---|
| **contact** | 1.000000 | -0.324315 | 0.019583 | -0.022840 | 0.077368 | 0.117970 | -0.212848 | |
| **month** | -0.324315 | 1.000000 | -0.006959 | -0.055570 | -0.030635 | -0.079556 | 0.063754 | |
| **day_of_week** | 0.019583 | -0.006959 | 1.000000 | -0.007520 | 0.015098 | 0.006765 | 0.004013 | - |
| **duration** | -0.022840 | -0.055570 | -0.007520 | 1.000000 | -0.132341 | -0.084064 | 0.058135 | |
| **campaign** | 0.077368 | -0.030635 | 0.015098 | -0.132341 | 1.000000 | 0.052584 | -0.079141 | |
| **pdays** | 0.117970 | -0.079556 | 0.006765 | -0.084064 | 0.052584 | 1.000000 | -0.587514 | - |
| **previous** | -0.212848 | 0.063754 | 0.004013 | 0.058135 | -0.079141 | -0.587514 | 1.000000 | - |
| **poutcome** | 0.118744 | 0.028950 | -0.012788 | 0.038291 | 0.032586 | -0.475619 | -0.313110 | |

In [279]:

```
sns.set_context("notebook",font_scale = 1.0, rc = {"lines.linewidth":2.5
plt.figure(figsize = (13,7))
a = sns.heatmap(corr, annot = True, fmt = ".2f")
```



## 3. Economic Data

In [280]:

```
# Slicing market economic index data
economicbankdata = bankdata[["emp.var.rate","cons.price.idx","cons.conf.i
economicbankdata.head()
```

Out[280]:

| | emp.var.rate | cons.price.idx | cons.conf.idx | euribor3m | nr.employed | y |
|---|---|---|---|---|---|---|
| 0 | 1.1 | 93.994 | -36.4 | 4.857 | 5191.0 | no |
| 1 | 1.1 | 93.994 | -36.4 | 4.857 | 5191.0 | no |
| 2 | 1.1 | 93.994 | -36.4 | 4.857 | 5191.0 | no |
| 3 | 1.1 | 93.994 | -36.4 | 4.857 | 5191.0 | no |
| 4 | 1.1 | 93.994 | -36.4 | 4.857 | 5191.0 | no |

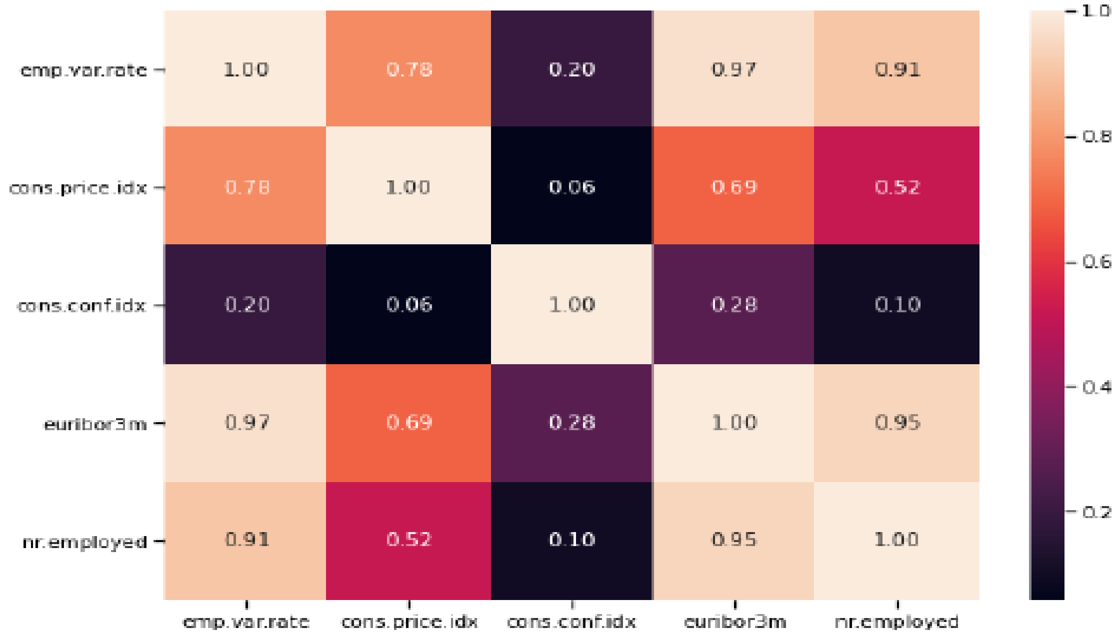
In [281]:

```
econ_corr=economicbankdata.corr()
econ_corr
```

Out[281]:

| | emp.var.rate | cons.price.idx | cons.conf.idx | euribor3m | nr.employed |
|---|---|---|---|---|---|
| emp.var.rate | 1.000000 | 0.775334 | 0.196041 | 0.972245 | 0.906970 |
| cons.price.idx | 0.775334 | 1.000000 | 0.058986 | 0.688230 | 0.522034 |
| cons.conf.idx | 0.196041 | 0.058986 | 1.000000 | 0.277686 | 0.100513 |
| euribor3m | 0.972245 | 0.688230 | 0.277686 | 1.000000 | 0.945154 |
| nr.employed | 0.906970 | 0.522034 | 0.100513 | 0.945154 | 1.000000 |

In [282]:
```
sns.set_context("notebook",font_scale = 1.0, rc = {"lines.linewidth":2.5
plt.figure(figsize = (10,7))
a = sns.heatmap(econ_corr, annot = True, fmt = ".2f")
```




In the above correlation , we can see that emp.var.rate is highly correlated with euribor3m & nr.employed variables so we can include these variables for our modelling.

***Combining 3 dataframes:***

At this stage, we are combining all these three data-frames to make a nice and clean version of the dataset which is ready for Featuring Engineering. After we have successfully created new features we would be able to research thoroughly as to what new feature is making the highest influence in term deposit subscription.

In [283]:
```
combinebankinfo= pd.concat([clientbankdata_copy1, bank_marketing, econom
combinebankinfo.shape
```

Out[283]: (41188, 48)

In [284]:
```
1 combinebankinfo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 48 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   age                            41188 non-null  int32
 1   job_admin.                     41188 non-null  uint8
 2   job_blue-collar                41188 non-null  uint8
 3   job_entrepreneur               41188 non-null  uint8
 4   job_housemaid                  41188 non-null  uint8
 5   job_management                 41188 non-null  uint8
 6   job_retired                    41188 non-null  uint8
 7   job_self-employed              41188 non-null  uint8
 8   job_services                   41188 non-null  uint8
 9   job_student                    41188 non-null  uint8
 10  job_technician                 41188 non-null  uint8
 11  job_unemployed                 41188 non-null  uint8
 12  job_unknown                    41188 non-null  uint8
 13  marital_divorced               41188 non-null  uint8
 14  marital_married                41188 non-null  uint8
 15  marital_single                 41188 non-null  uint8
 16  marital_unknown                41188 non-null  uint8
 17  education_basic.4y             41188 non-null  uint8
 18  education_basic.6y             41188 non-null  uint8
 19  education_basic.9y             41188 non-null  uint8
 20  education_high.school          41188 non-null  uint8
 21  education_illiterate           41188 non-null  uint8
 22  education_professional.course  41188 non-null  uint8
 23  education_university.degree    41188 non-null  uint8
 24  education_unknown              41188 non-null  uint8
 25  default_no                     41188 non-null  uint8
 26  default_unknown                41188 non-null  uint8
 27  default_yes                    41188 non-null  uint8
 28  housing_no                     41188 non-null  uint8
 29  housing_unknown                41188 non-null  uint8
 30  housing_yes                    41188 non-null  uint8
 31  loan_no                        41188 non-null  uint8
 32  loan_unknown                   41188 non-null  uint8
 33  loan_yes                       41188 non-null  uint8
 34  contact                        41188 non-null  int32
 35  month                          41188 non-null  int64
 36  day_of_week                    41188 non-null  int64
 37  duration                       41188 non-null  float64
 38  campaign                       41188 non-null  int64
 39  pdays                          41188 non-null  int64
 40  previous                       41188 non-null  int64
 41  poutcome                       41188 non-null  int32
 42  emp.var.rate                   41188 non-null  float64
 43  cons.price.idx                 41188 non-null  float64
 44  cons.conf.idx                  41188 non-null  float64
 45  euribor3m                      41188 non-null  float64
 46  nr.employed                    41188 non-null  float64
 47  y                              41188 non-null  object
dtypes: float64(6), int32(3), int64(5), object(1), uint8(33)
memory usage: 5.5+ MB
```

In [285]:

```
1 bankcorr = combinebankinfo.corr()
2 bankcorr
```

Out[285]:

| | age | job_admin. | job_blue-collar | job_entrepreneur | job_housem |
|---|---|---|---|---|---|
| age | 1.000000 | -0.090483 | 0.013375 | 0.044661 | 0.0802 |
| job_admin. | -0.090483 | 1.000000 | -0.313313 | -0.111417 | -0.0945 |
| job_blue-collar | 0.013375 | -0.313313 | 1.000000 | -0.103050 | -0.0874 |
| job_entrepreneur | 0.044661 | -0.111417 | -0.103050 | 1.000000 | -0.0311 |
| job_housemaid | 0.080294 | -0.094595 | -0.087492 | -0.031113 | 1.0000 |
| job_management | 0.078006 | -0.160892 | -0.148810 | -0.052918 | -0.0449 |
| job_retired | 0.232925 | -0.121502 | -0.112378 | -0.039962 | -0.0339 |
| job_self-employed | 0.006874 | -0.110021 | -0.101759 | -0.036186 | -0.0307 |
| job_services | -0.049218 | -0.190063 | -0.175791 | -0.062513 | -0.0530 |
| job_student | -0.176706 | -0.085748 | -0.079308 | -0.028203 | -0.0239 |
| job_technician | -0.053919 | -0.257516 | -0.238178 | -0.084698 | -0.0719 |
| job_unemployed | 0.007569 | -0.092467 | -0.085523 | -0.030413 | -0.0258 |
| job_unknown | 0.041439 | -0.052307 | -0.048379 | -0.017204 | -0.0146 |
| marital_divorced | 0.155294 | 0.020013 | -0.056857 | 0.006657 | 0.0205 |
| marital_married | 0.288075 | -0.120494 | 0.129272 | 0.051050 | 0.0424 |
| marital_single | -0.422213 | 0.117787 | -0.100192 | -0.060245 | -0.0609 |
| marital_unknown | -0.000706 | -0.007918 | -0.005251 | 0.000514 | 0.0032 |
| education_basic.4y | 0.198560 | -0.181255 | 0.265906 | -0.004627 | 0.1861 |
| education_basic.6y | 0.026371 | -0.104499 | 0.231184 | -0.005748 | 0.0120 |
| education_basic.9y | -0.017342 | -0.162641 | 0.372303 | -0.001371 | -0.0266 |
| education_high.school | -0.090406 | 0.122080 | -0.173873 | -0.031929 | -0.0257 |
| education_illiterate | 0.015639 | -0.009498 | 0.011010 | 0.008579 | 0.0039 |
| education_professional.course | 0.005940 | -0.161464 | -0.126531 | -0.019858 | -0.0349 |
| education_university.degree | -0.080180 | 0.327321 | -0.336592 | 0.051832 | -0.0585 |
| education_unknown | 0.062479 | -0.052604 | 0.018869 | -0.002746 | -0.0019 |
| default_no | -0.197102 | 0.121336 | -0.176579 | 0.000974 | -0.0368 |
| default_unknown | 0.197038 | -0.121248 | 0.176698 | -0.000940 | 0.0368 |
| default_yes | 0.004260 | -0.004967 | -0.004594 | -0.001634 | -0.0013 |
| housing_no | 0.004057 | -0.008529 | 0.014033 | -0.004567 | 0.0036 |
| housing_unknown | -0.000489 | -0.008570 | 0.006673 | 0.000861 | 0.0035 |
| housing_yes | -0.003893 | 0.011128 | -0.016031 | 0.004287 | -0.0046 |
| loan_no | 0.005401 | -0.015485 | 0.003089 | 0.004789 | 0.0013 |
| loan_unknown | -0.000489 | -0.008570 | 0.006673 | 0.000861 | 0.0035 |

In [286]:

```
1 bankcorr.to_csv('bankcorr.csv')
```

As shown below,we're representing top absolute correlation features which will acount towards term deposit

In [287]:

```
def get_redundant_pairs(bankcorr):
    '''Get diagonal and lower triangular pairs of correlation matrix'''
    pairs_to_drop = set()
    cols = bankcorr.columns
    for i in range(0, bankcorr.shape[1]):
        for j in range(0, i+1):
            pairs_to_drop.add((cols[i], cols[j]))
    return pairs_to_drop

def get_top_abs_correlations(bankcorr, n=5):
    au_corr = bankcorr.corr().abs().unstack()
    labels_to_drop = get_redundant_pairs(bankcorr)
    au_corr = au_corr.drop(labels=labels_to_drop).sort_values(ascending=
    return au_corr[0:n]

print("Top Absolute Correlations")
print(get_top_abs_correlations(bankcorr, 50))
```

```
Top Absolute Correlations
housing_unknown    loan_unknown                     1.000000
default_no         default_unknown                  0.999908
emp.var.rate       euribor3m                        0.995691
euribor3m          nr.employed                      0.987471
emp.var.rate       nr.employed                      0.981645
housing_no         housing_yes                      0.949334
emp.var.rate       cons.price.idx                   0.925485
loan_no            loan_yes                         0.913321
cons.price.idx     euribor3m                        0.895179
marital_married    marital_single                   0.860736
contact            cons.price.idx                   0.858262
cons.price.idx     nr.employed                      0.846535
previous           nr.employed                      0.813394
                   euribor3m                        0.779324
job_technician     education_professional.course    0.775783
previous           emp.var.rate                     0.766842
pdays              previous                         0.743261
age                marital_single                   0.733898
contact            emp.var.rate                     0.722860
                   euribor3m                        0.701706
                   nr.employed                      0.633739
previous           cons.price.idx                   0.626563
job_blue-collar    education_basic.9y               0.624887
pdays              nr.employed                      0.620679
job_services       education_high.school            0.600964
age                marital_married                  0.572005
job_admin.         education_university.degree      0.561873
contact            previous                         0.555161
job_blue-collar    education_university.degree      0.553858
pdays              euribor3m                        0.547807
                   emp.var.rate                     0.534491
job_blue-collar    education_basic.4y               0.470065
job_admin.         job_blue-collar                  0.465718
loan_no            loan_unknown                     0.457061
housing_unknown    loan_no                          0.457061
job_student        marital_single                   0.456990
contact            month                            0.442467
```