In [1]:

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

In [4]:

```
mydata=pd.read_csv("bank-additional.csv")
pd.set_option("display.max_columns",50)
```

In [5]:

```
mydata
```

Out[5]:

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

41188 rows × 21 columns

In [6]:

```
mydata.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 [7]:

```
mydata.describe()
```

Out[7]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	411
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	
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	
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	

In [8]:

```
mydata.isnull().sum()
```

Out[8]:

```
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 [9]:

```
mydata_corr=mydata.corr()
```

In [10]:

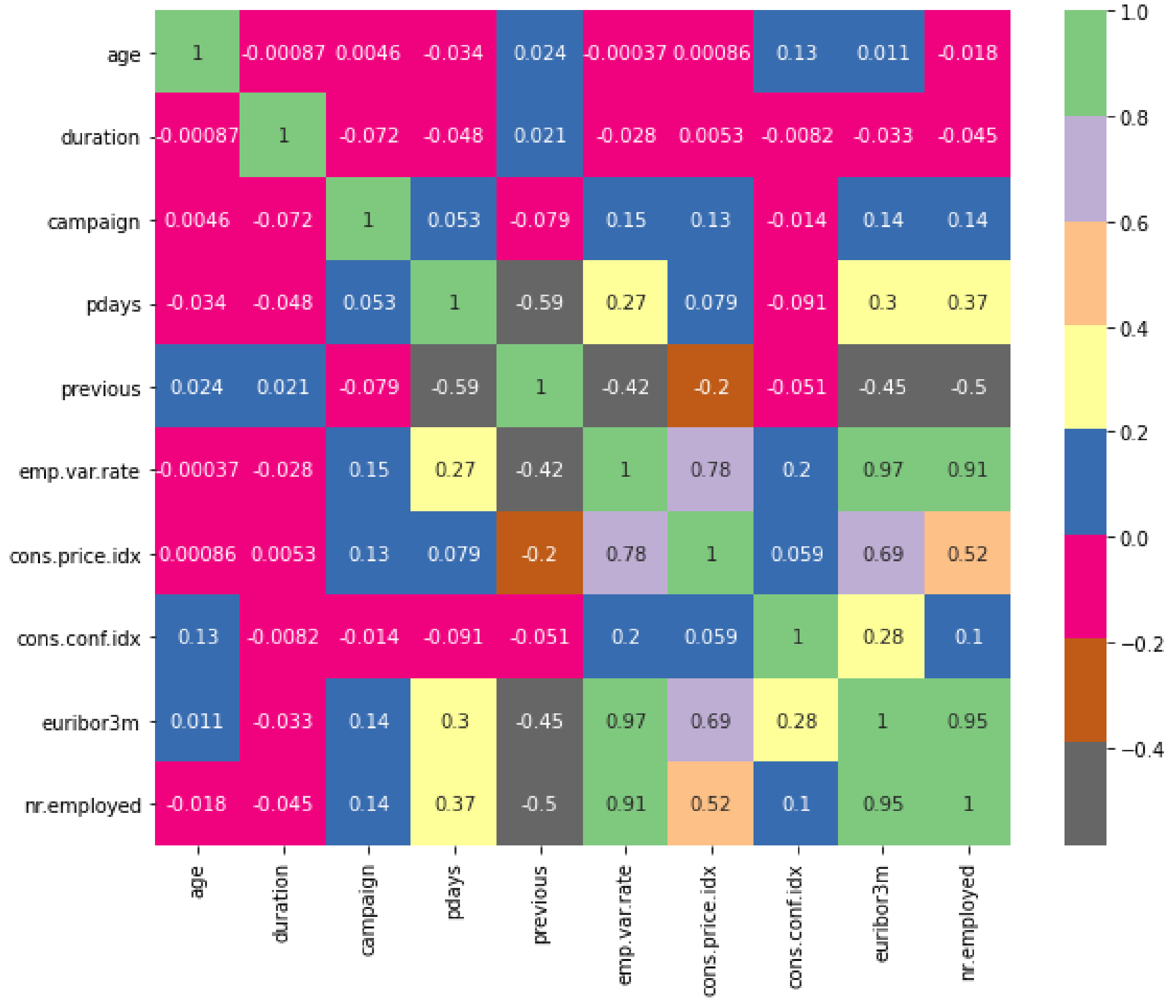
```
mydata_corr
```

Out[10]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx
age	1.000000	-0.000866	0.004594	-0.034369	0.024365	-0.000371	0.000857
duration	-0.000866	1.000000	-0.071699	-0.047577	0.020640	-0.027968	0.005312
campaign	0.004594	-0.071699	1.000000	0.052584	-0.079141	0.150754	0.127836
pdays	-0.034369	-0.047577	0.052584	1.000000	-0.587514	0.271004	0.078889
previous	0.024365	0.020640	-0.079141	-0.587514	1.000000	-0.420489	-0.203130
emp.var.rate	-0.000371	-0.027968	0.150754	0.271004	-0.420489	1.000000	0.775334
cons.price.idx	0.000857	0.005312	0.127836	0.078889	-0.203130	0.775334	1.000000
cons.conf.idx	0.129372	-0.008173	-0.013733	-0.091342	-0.050936	0.196041	0.058986
euribor3m	0.010767	-0.032897	0.135133	0.296899	-0.454494	0.972245	0.688230
nr.employed	-0.017725	-0.044703	0.144095	0.372605	-0.501333	0.906970	0.522034

In [16]:

```
plt.figure(figsize=(10,8))
sns.heatmap(mydata_corr,annot=True,cmap='Accent_r')
plt.show()
```

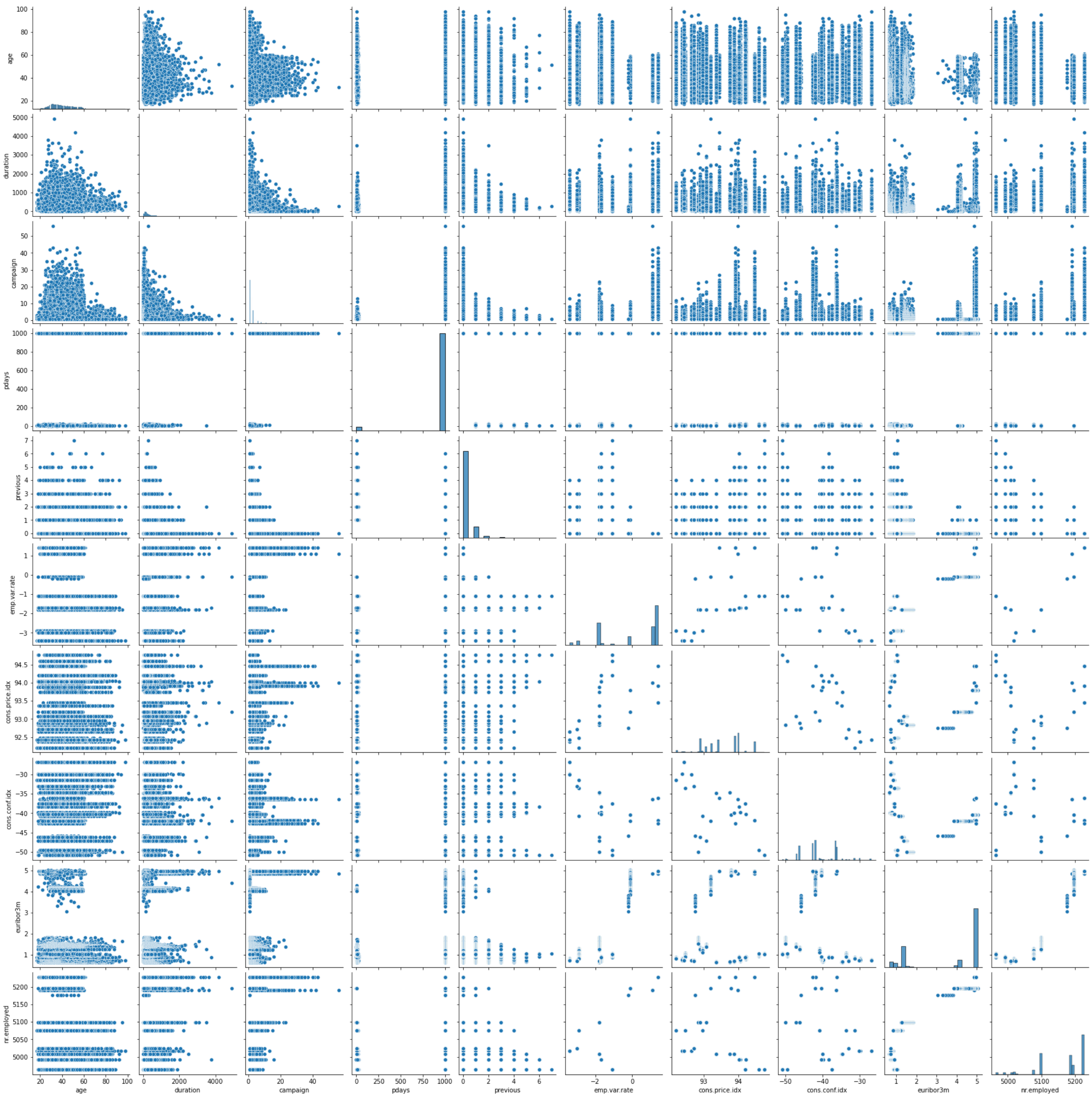


In [17]:

```
sns.pairplot(mydata)
```

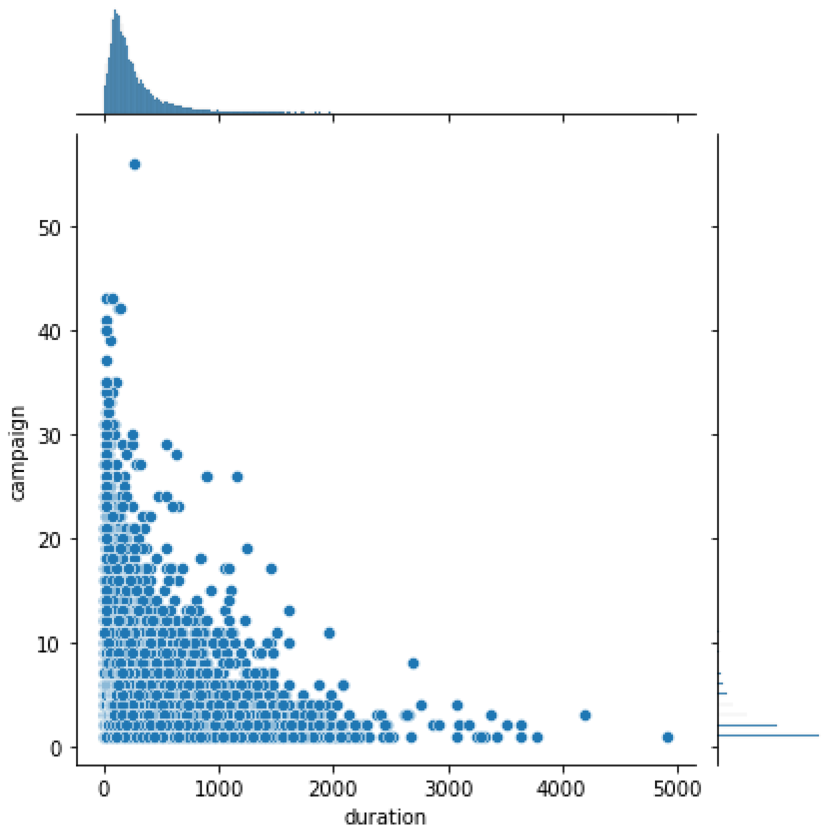
Out[17]:

<seaborn.axisgrid.PairGrid at 0x1ef247b7550>



In [19]:

```
sns.jointplot(x='duration',y='campaign',data=mydata);
```



In [20]:

```
from sklearn.preprocessing import LabelEncoder
```

In [22]:

```
LE = LabelEncoder()
for col in mydata.select_dtypes(include=['object']).columns.values:
    mydata[col] = LE.fit_transform(mydata[col])
```

In [23]:

```
mydata
```

Out[23]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	dura
0	56	3	1	0	0	0	0	1	6	1	
1	57	7	1	3	1	0	0	1	6	1	
2	37	7	1	3	0	2	0	1	6	1	
3	40	0	1	1	0	0	0	1	6	1	
4	56	7	1	3	0	0	2	1	6	1	
...	...	...	...	...	...	...	...	...	...	...	
41183	73	5	1	5	0	2	0	0	7	0	
41184	46	1	1	5	0	0	0	0	7	0	
41185	56	5	1	6	0	2	0	0	7	0	
41186	44	9	1	5	0	0	0	0	7	0	
41187	74	5	1	5	0	2	0	0	7	0	

41188 rows × 21 columns

In [24]:

```
y_dep=mydata.y
```

In [25]:

```
y_dep
```

Out[25]:

```
0        0
1        0
2        0
3        0
4        0
        ..
41183    1
41184    0
41185    0
41186    1
41187    0
Name: y, Length: 41188, dtype: int32
```

In [26]:

```
x_ind=mydata.drop("y",axis=1)
```

In [27]:

```
x_ind
```

Out[27]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	dura
0	56	3	1	0	0	0	0	1	6	1	
1	57	7	1	3	1	0	0	1	6	1	
2	37	7	1	3	0	2	0	1	6	1	
3	40	0	1	1	0	0	0	1	6	1	
4	56	7	1	3	0	0	2	1	6	1	
...	...	...	...	...	...	...	...	...	...	...	
41183	73	5	1	5	0	2	0	0	7	0	
41184	46	1	1	5	0	0	0	0	7	0	
41185	56	5	1	6	0	2	0	0	7	0	
41186	44	9	1	5	0	0	0	0	7	0	
41187	74	5	1	5	0	2	0	0	7	0	

41188 rows × 20 columns

In [28]:

```
from sklearn.model_selection import train_test_split
```

In [29]:

```
x_train,x_test,y_train,y_test=train_test_split(x_ind,y_dep,train_size=0.2,random_state=2)
```

In [30]:

```
from sklearn.preprocessing import StandardScaler
```

In [31]:

```
norm=StandardScaler()
```

In [32]:

```
x_train=norm.fit_transform(x_train)
```

In [33]:

```
x_test=norm.fit_transform(x_test)
```

Model Building

In [34]:

```
from sklearn.svm import SVC
```

In [35]:

```
model_svc=SVC(kernel='linear')
```

In [36]:

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

Out[36]:

```
SVC(kernel='linear')
```

In [37]:

```
y_pred_svc=model_svc.predict(x_test)
```

In [38]:

```
y_pred_svc
```

Out[38]:

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

Confusion Matrix and Accuracy Score

In [39]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [40]:

```
confusion_matrix(y_test,y_pred_svc)
```

Out[40]:

```
array([[28604,   620],
       [ 2424,  1303]], dtype=int64)
```

In [41]:

```
accuracy_score(y_test,y_pred_svc)*100
```

Out[41]:

```
90.76204060574793
```

In [42]:

```
model_svc.n_support_
```

Out[42]:

```
array([836, 826])
```

In [43]:

```
clas_report=classification_report(y_test,y_pred_svc)
print(clas_report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95     29224
           1       0.68      0.35      0.46      3727

    accuracy                           0.91     32951
   macro avg       0.80      0.66      0.71     32951
weighted avg       0.89      0.91      0.89     32951
```

In [44]:

```
kernel=('linear','rbf','poly','sigmoid')
```

In [46]:

```
for i in kernel:
    model1=SVC(kernel=i)
    model1=model1.fit(x_train,y_train)
    print("Kernel: ", i)
    print("Accuracy score: ", accuracy_score(y_test,model1.predict(x_test)))
```

```
Kernel:  linear
Accuracy score:  0.9076204060574793
Kernel:  rbf
Accuracy score:  0.9068920518345422
Kernel:  poly
Accuracy score:  0.9031288883493672
Kernel:  sigmoid
Accuracy score:  0.8786986737883524
```

In [ ]:

```

```