In [129]:

```
import pandas as pd
import numpy as np
import math
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
```

In [130]:

```
bank=pd.read_csv('bank-full.csv',sep=";")
```

In [131]:

```
bank.head()
```

Out[131]:

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

In [132]:

```
print("{rows}".format(rows = len(bank)))
```

```
45211
```

In [133]:

```
bank.isnull().sum()
```

Out[133]:

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [134]:

```
bank.describe()
```

Out[134]:

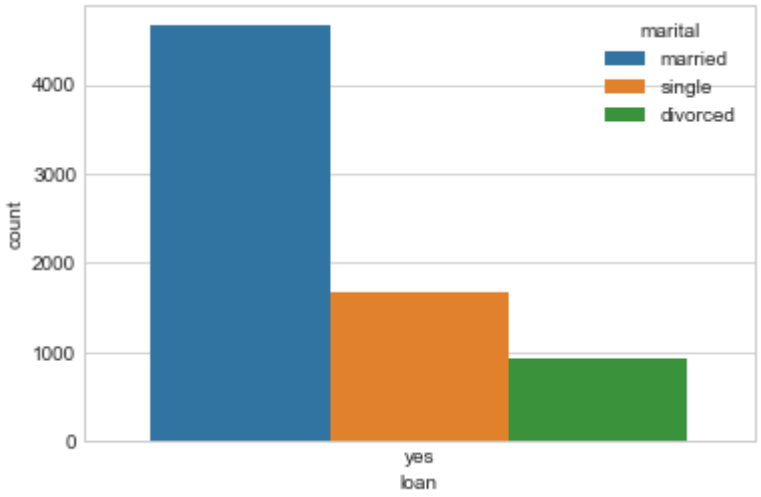
	age	balance	day	duration	campaign	pdays	45
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	
min	18.000000	-8019.000000	1.000000	0.000000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

In [136]:

```
sns.countplot(x=bank[bank['loan']=="yes"]['loan'],hue='marital',data=bank)
```

Out[136]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd98c1c898>
```

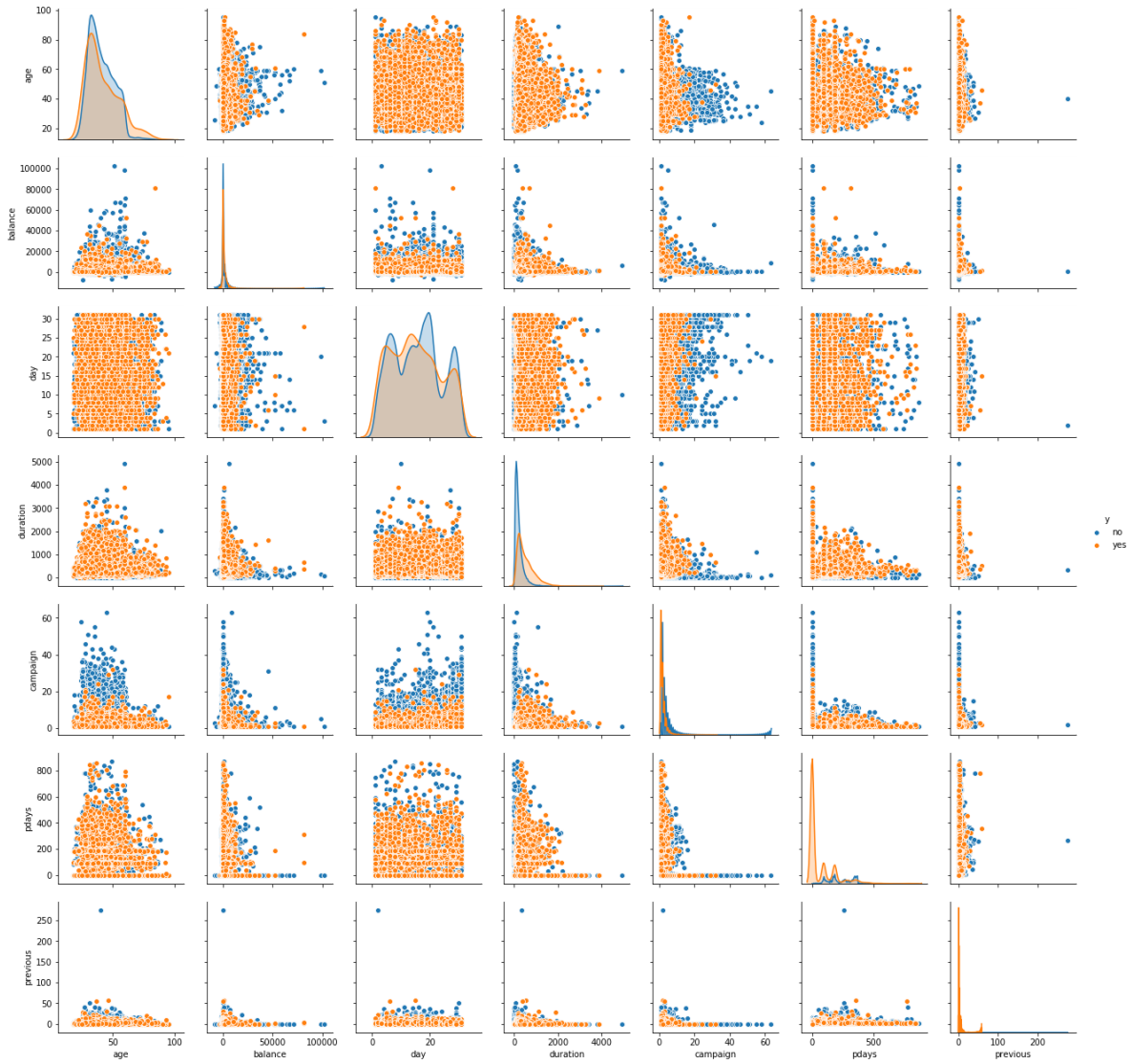


In [17]:

```
sns.pairplot(bank,hue='y')
```

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x1dd91521908>
```

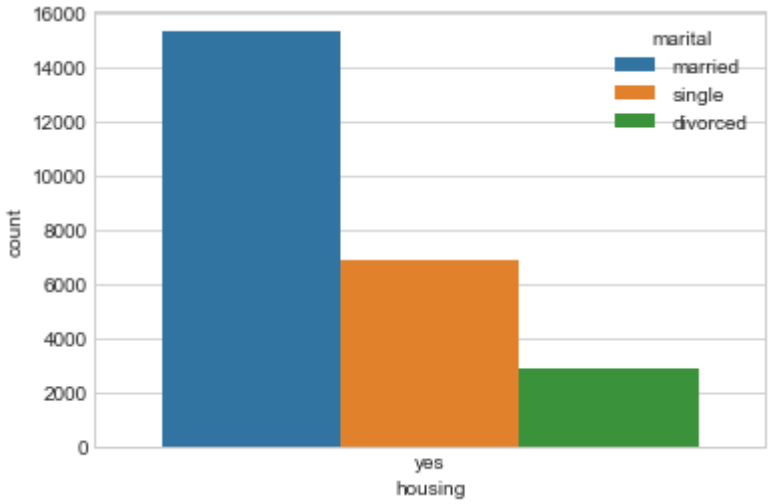


In [137]:

```
sns.countplot(x=bank[bank['housing']=="yes"]['housing'],hue='marital',data=bank)
```

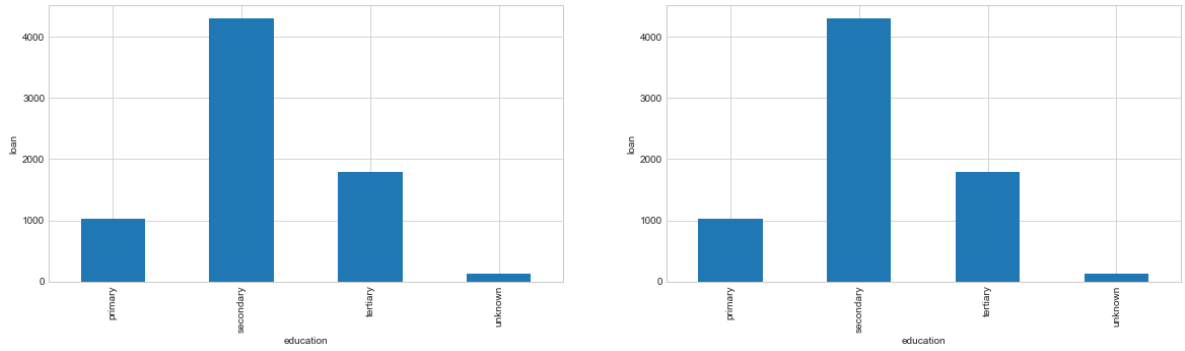
Out[137]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd9805e278>
```



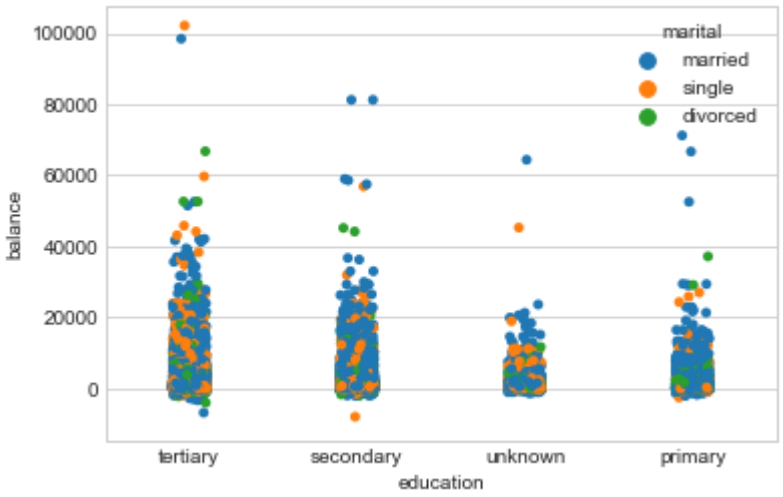
In [162]:

```
data1=bank.groupby('education').apply(lambda x:(x[x['loan']=="yes"]['loan']).count())
plt.subplot(1,2,1)
data1.plot(kind='bar' , figsize= (20,5))
plt.ylabel("loan")

#print(data1)
data2=bank.groupby('education').apply(lambda x:(x[x['loan']=="yes"]['loan']).count())
plt.subplot(1,2,2)
data2.plot(kind='bar')
plt.ylabel("loan")
#print(data1)

plt.show()
```



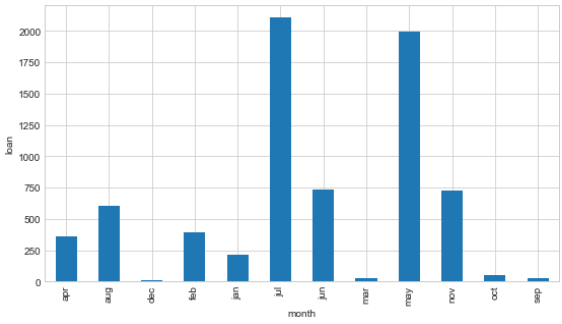
In [139]:

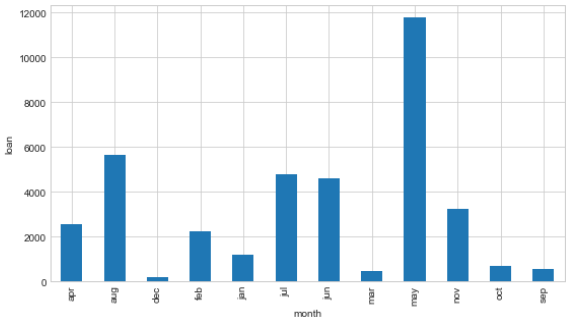
```
ax=sns.stripplot(x="education",y="balance",hue="marital",data=bank)
```



In [164]:

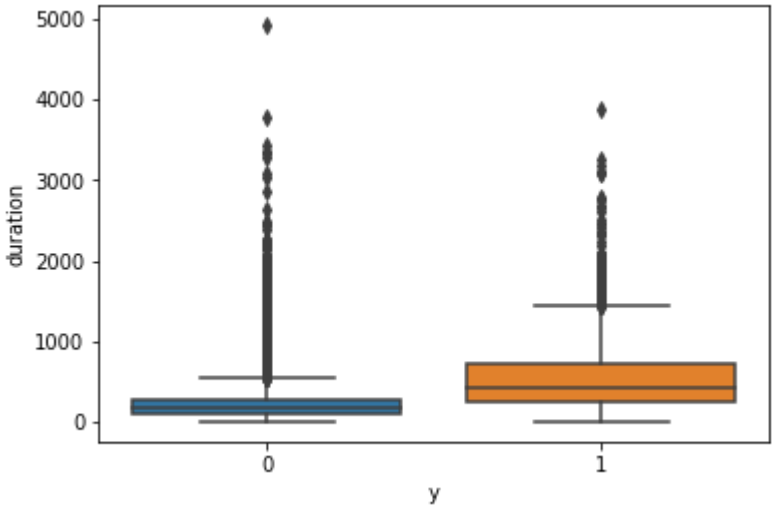
```
data1=bank.groupby('month').apply(lambda x:(x[x['loan']=="yes"]['loan']).count())
plt.subplot(1,2,1)
data1.plot(kind='bar' , figsize=(20,5))
plt.ylabel("loan")
data2=bank.groupby('month').apply(lambda x:(x[x['loan']=="no"]['loan']).count())
plt.subplot(1,2,2)
data2.plot(kind='bar')
plt.ylabel("loan")
#print(data2)
plt.show()
```





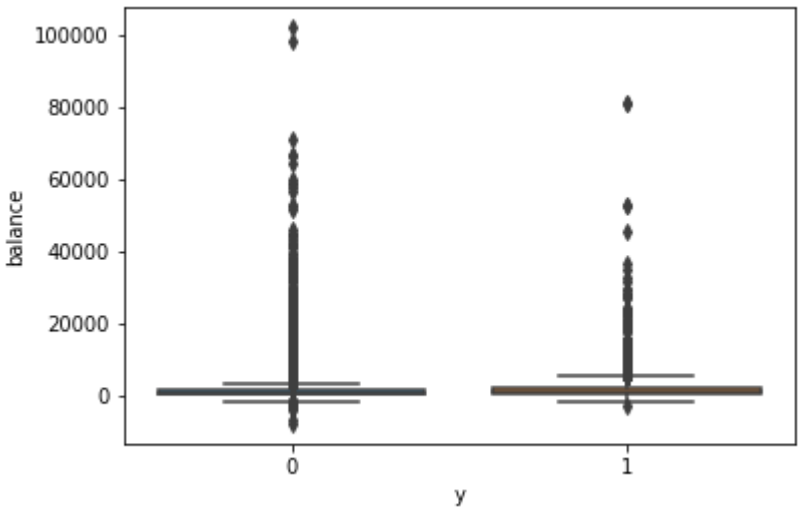
In [127]:

```
ax = sns.boxplot(y=bank["duration"], x = bank['y'])
```



In [116]:

```
ax = sns.boxplot(y=bank["balance"], x = bank['y'])
```

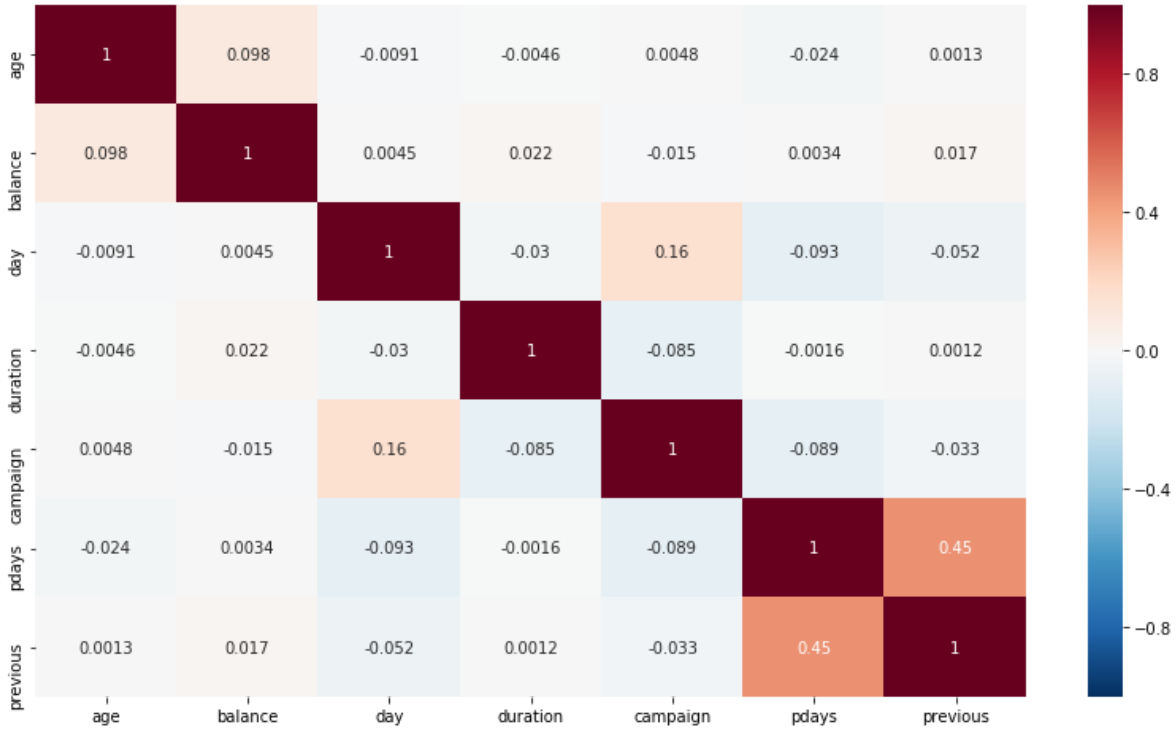


In [13]:

```
correlation=bank.corr()
plt.figure(figsize=(14,8))
sns.heatmap(correlation,annot=True,linewidth=0,vmin=-1,cmap="RdBu_r")
```

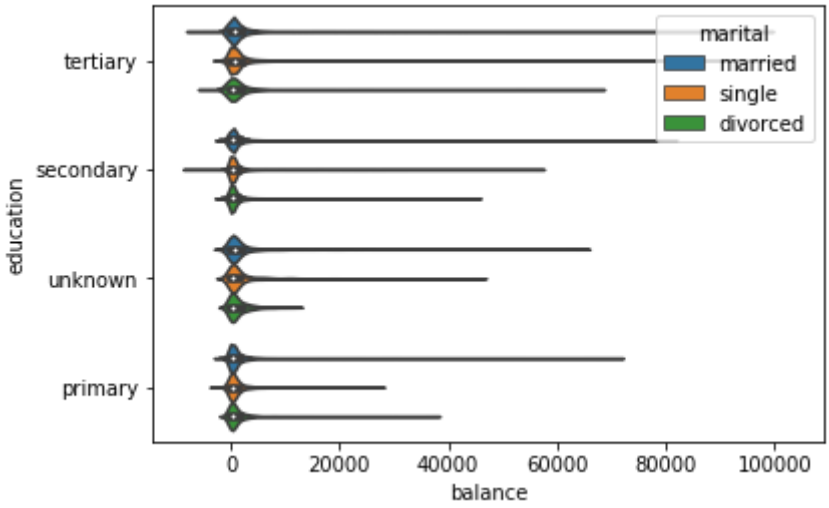
Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd91b22e10>
```



In [15]:

```
ax=sns.violinplot(x="balance",y="education",hue="marital",data=bank)
```



In [61]:

```
label_encoder = LabelEncoder()
bank['job'] = label_encoder.fit_transform(bank['job'])
bank['marital'] =label_encoder.fit_transform(bank['marital'])
bank['education'] = label_encoder.fit_transform(bank['education'])
bank['default'] = label_encoder.fit_transform(bank['default'])
bank['housing'] = label_encoder.fit_transform(bank['housing'])
bank['loan'] = label_encoder.fit_transform(bank['loan'])
bank['month'] = label_encoder.fit_transform(bank['month'])
#bank['day_of_week'] = label_encoder.fit_transform(bank['day_of_week'])
bank['poutcome'] = label_encoder.fit_transform(bank['poutcome'])
bank['y'] = label_encoder.fit_transform(bank['y'])
```

In [62]:

```
bank.dtypes
```

Out[62]:

```
age           int64
job           int32
marital       int32
education     int32
default       int32
balance       int64
housing       int32
loan          int32
contact      object
day           int64
month         int32
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome      int32
y             int32
dtype: object
```

In [63]:

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

In [64]:

```
bank['contact'] = label_encoder.fit_transform(bank['contact'])
```

In [65]:

```
def Age(ag):
    if ag > 60 : return 1
    elif 60 >ag >=45:
        return 2
    elif 45 > ag >=30:
        return 3
    elif 30 > ag >=15 :
        return 4
    else :
        return 5
bank['age']=bank['age'].map(Age)
```

In [90]:

```
def Pdays(pd):
    if pd == 871 : return 1

    else :
        return 0
bank['pdays']=bank['pdays'].map(Pdays)
```

In [91]:

```
bank.head()
```

Out[91]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
0	2	4	1	2	0	2143	1	0	2	5	8	261
1	3	9	2	1	0	29	1	0	2	5	8	151
2	3	2	1	1	0	2	1	1	2	5	8	76
3	2	1	1	3	0	1506	1	0	2	5	8	92
4	3	11	2	3	0	1	0	0	2	5	8	198

In [67]:

```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
age          45211 non-null int64
job          45211 non-null int32
marital      45211 non-null int32
education    45211 non-null int32
default      45211 non-null int32
balance      45211 non-null int64
housing      45211 non-null int32
loan         45211 non-null int32
contact      45211 non-null int32
day          45211 non-null int64
month        45211 non-null int32
duration     45211 non-null int64
campaign     45211 non-null int64
pdays        45211 non-null int64
previous     45211 non-null int64
poutcome     45211 non-null int32
y            45211 non-null int32
dtypes: int32(10), int64(7)
memory usage: 4.1 MB
```

In [68]:

```
bank.groupby('age').y.value_counts()
```

Out[68]:

```
age  y
1    0      686
     1      502
2    0    12575
     1     1305
3    0    21818
     1     2456
4    0     4345
     1      928
5    0      498
     1       98
Name: y, dtype: int64
```

In [89]:

```
bank.groupby('loan').y.value_counts()
```

Out[89]:

```
loan  y
0     0    33162
      1     4805
1     0     6760
      1      484
Name: y, dtype: int64
```

In [69]:

```
bank.groupby('pdays').y.value_counts()
```

Out[69]:

```
pdays  y
0      0    39922
       1     5289
Name: y, dtype: int64
```

In [92]:

```
bank = bank.drop('poutcome', axis=1)
```

In [93]:

```
x = bank.drop('y', axis=1)
Y = bank['y']

X_train, X_test, y_train, y_test = train_test_split(x,Y,test_size=0.30,random_state=11)
```

In [153]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [95]:

```
model =RandomForestClassifier(n_estimators = 1000 , criterion = 'entropy' , random_state=3)
model.fit(X_train , y_train)
predicted = model.predict(X_test)
```

In [97]:

```
from sklearn import metrics

print('Accuracy:',round(metrics.accuracy_score(y_test,predicted),5))
```

```
Accuracy: 0.90077
```

In [74]:

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

In [75]:

```
results = confusion_matrix(y_test, predicted)
print(results)
print( classification_report(y_test, predicted))
```

```
[[11642   339]
 [  940   643]]
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     11981
           1       0.65      0.41      0.50      1583

   micro avg       0.91      0.91      0.91     13564
   macro avg       0.79      0.69      0.72     13564
weighted avg       0.89      0.91      0.90     13564
```

In [76]:

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

In [154]:

```
model =  KNeighborsClassifier(n_neighbors=3)
model.fit(X_train , y_train )
predicted2 = model.predict(X_test)
```

In [155]:

```
print('Accuracy:',round(metrics.accuracy_score(y_test,predicted2),5))
```

```
Accuracy: 0.86531
```

In [156]:

```
results = confusion_matrix(y_test, predicted2)
print(results)
print( classification_report(y_test, predicted2))
```

```
[[11367   614]
 [ 1213   370]]
              precision    recall  f1-score   support

           0       0.90      0.95      0.93     11981
           1       0.38      0.23      0.29      1583

   micro avg       0.87      0.87      0.87     13564
   macro avg       0.64      0.59      0.61     13564
weighted avg       0.84      0.87      0.85     13564
```

In [159]:

```
from sklearn.tree import DecisionTreeClassifier
model =  DecisionTreeClassifier()
model.fit(X_train , y_train )
predicted3 = model.predict(X_test)
```

In [160]:

```
print('Accuracy:',round(metrics.accuracy_score(y_test,predicted3),5))
```

```
Accuracy: 0.86523
```

In [161]:

```
results = confusion_matrix(y_test, predicted3)
print(results)
print( classification_report(y_test, predicted3))
```

```
[[11014   967]
 [  861   722]]
              precision    recall  f1-score   support

           0       0.93      0.92      0.92     11981
           1       0.43      0.46      0.44      1583

   micro avg       0.87      0.87      0.87     13564
   macro avg       0.68      0.69      0.68     13564
weighted avg       0.87      0.87      0.87     13564
```

In [ ]:

```

```