my-project-teachnook

June 19, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df=pd.read_csv("/content/bank-full (1).csv")
```

[3]:
```
df.head()
```

[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

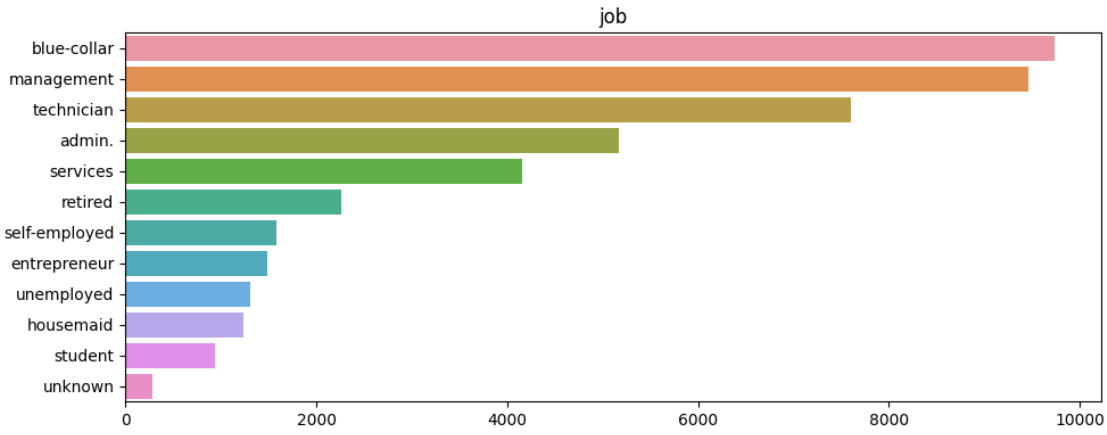
[4]:
```
print("shape:",df.shape)
print("========================================")
print("info:",df.info())
print("========================================")
print(df.describe())
print("========================================")
print("missing values:",df.isnull().sum())
```

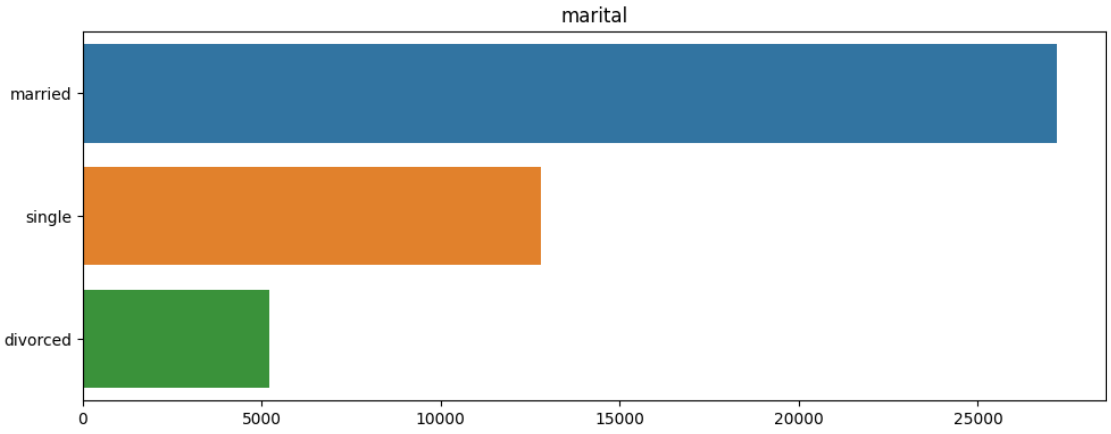
```
shape: (45211, 17)
========================================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
```

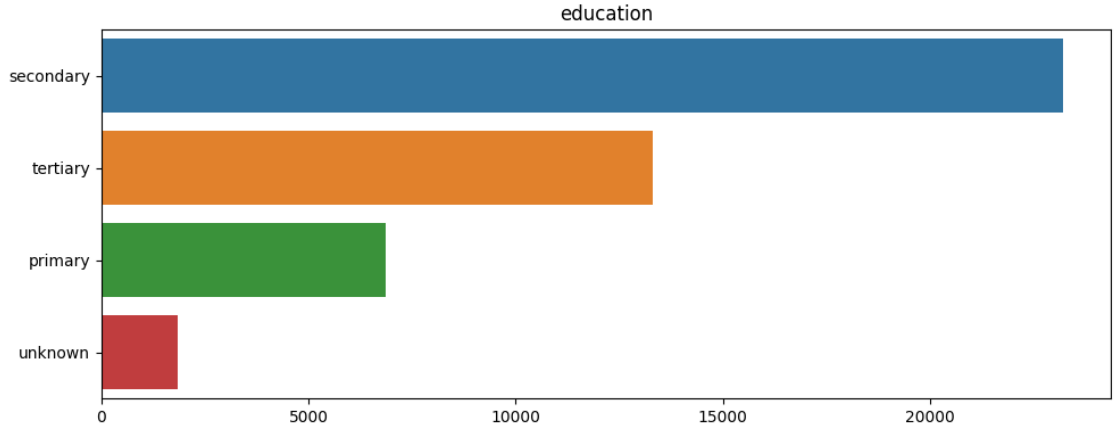
```
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
info: None
=======================================
                age        balance           day      duration      campaign  \
count  45211.000000   45211.000000  45211.000000  45211.000000  45211.000000
mean      40.936210    1362.272058     15.806419    258.163080      2.763841
std       10.618762    3044.765829      8.322476    257.527812      3.098021
min       18.000000   -8019.000000      1.000000      0.000000      1.000000
25%       33.000000      72.000000      8.000000    103.000000      1.000000
50%       39.000000     448.000000     16.000000    180.000000      2.000000
75%       48.000000    1428.000000     21.000000    319.000000      3.000000
max       95.000000  102127.000000     31.000000   4918.000000     63.000000

              pdays      previous
count  45211.000000  45211.000000
mean      40.197828      0.580323
std      100.128746      2.303441
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       -1.000000      0.000000
max      871.000000    275.000000
=======================================
missing values: age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
```

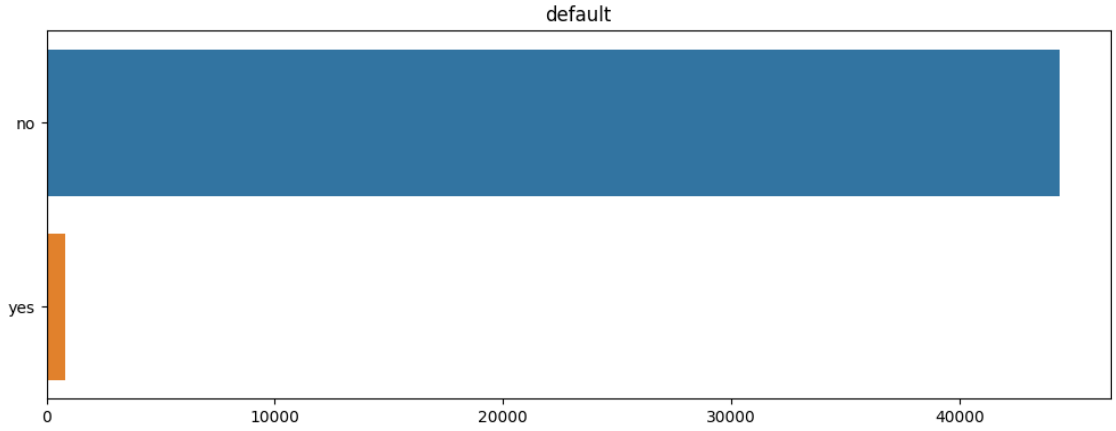
```
loan          0
contact       0
day           0
month         0
duration      0
campaign      0
pdays         0
previous      0
poutcome      0
y             0
dtype: int64
```

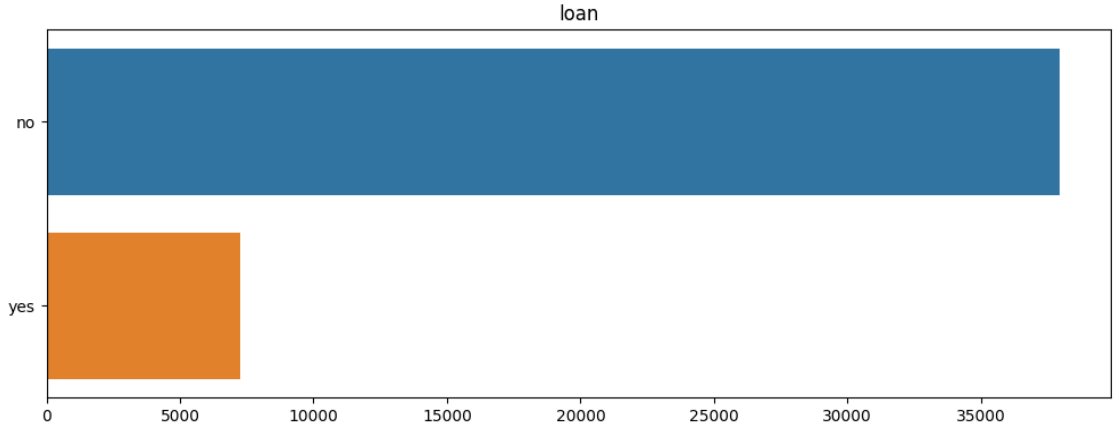
[12]:
```
categorcial_variables = ['job', 'marital', 'education', 'default', 'loan',␣
 ↪'contact', 'month', 'day', 'poutcome','y']
for col in categorcial_variables:
    plt.figure(figsize=(10,4))
    sns.barplot(x=df[col].value_counts().values, y=df[col].value_counts().index)
    plt.title(col)
    plt.tight_layout()
```

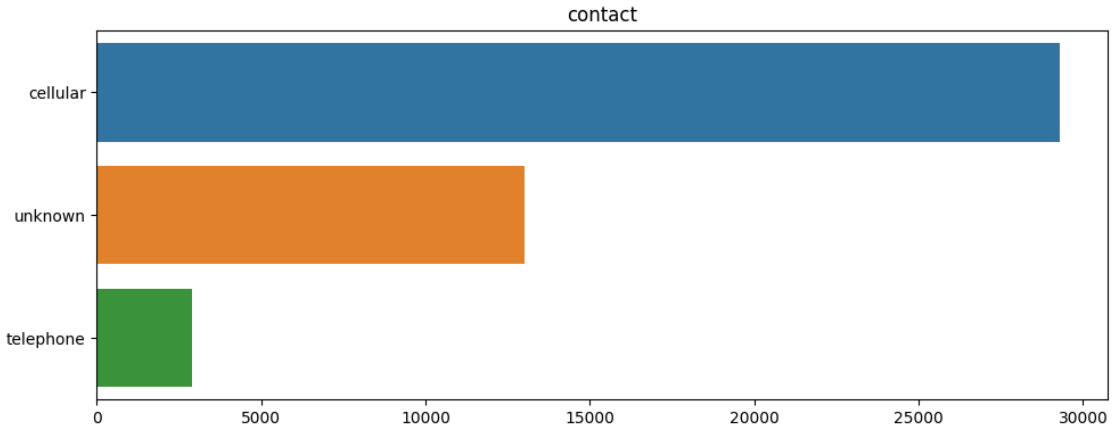


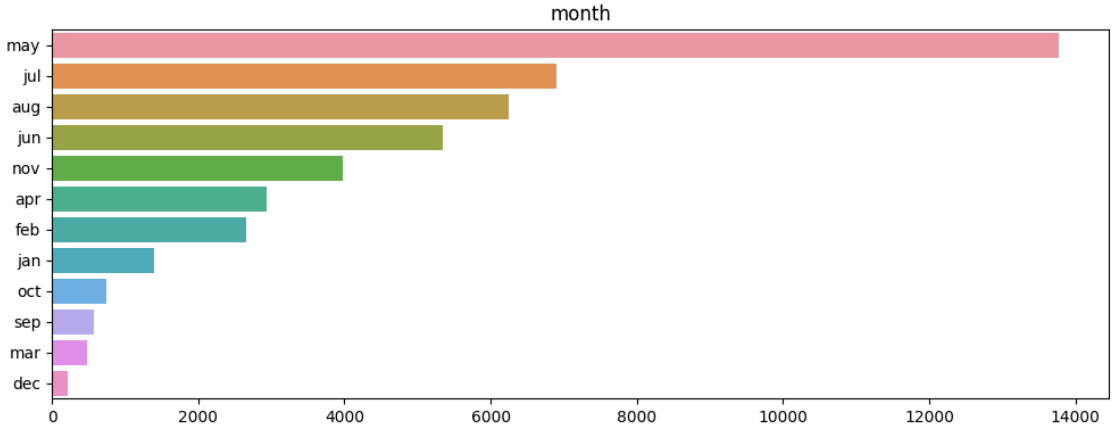


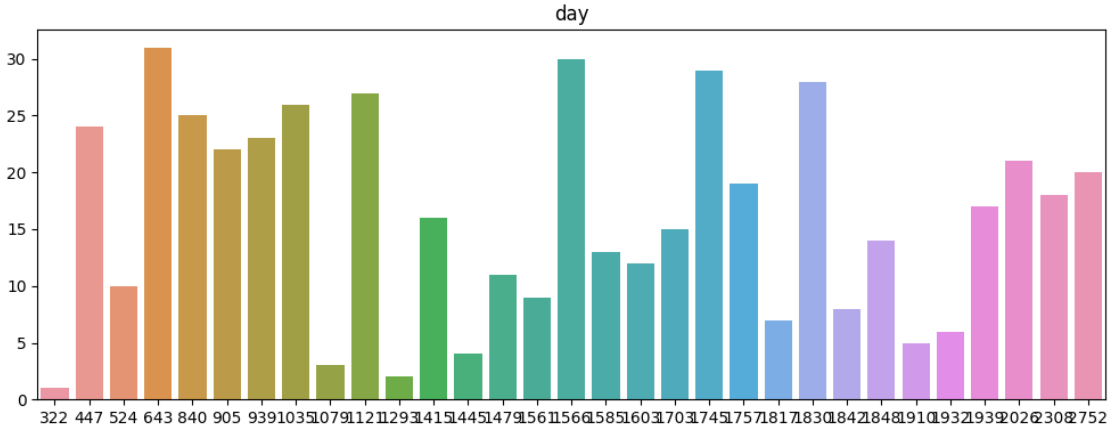

education
secondary
tertiary
primary
unknown
0
5000
10000
15000
20000

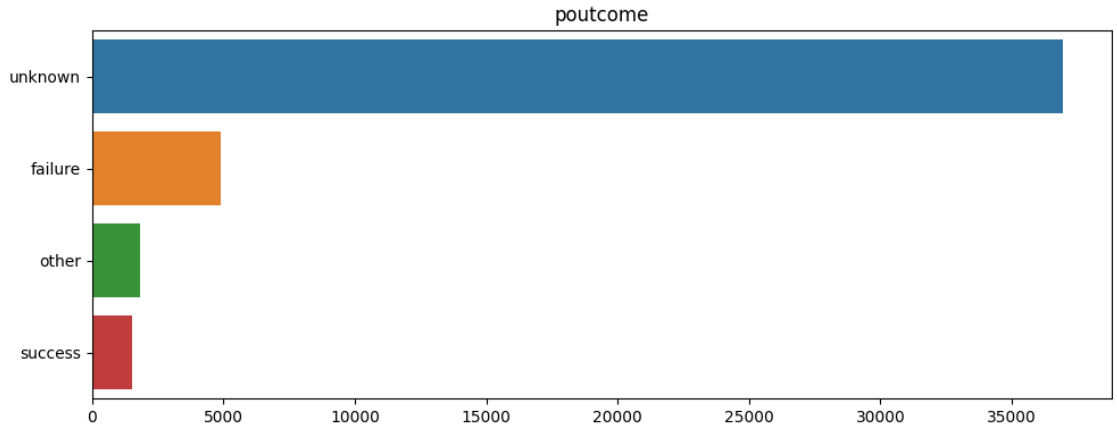

default
no
yes
0
10000
20000
30000
40000

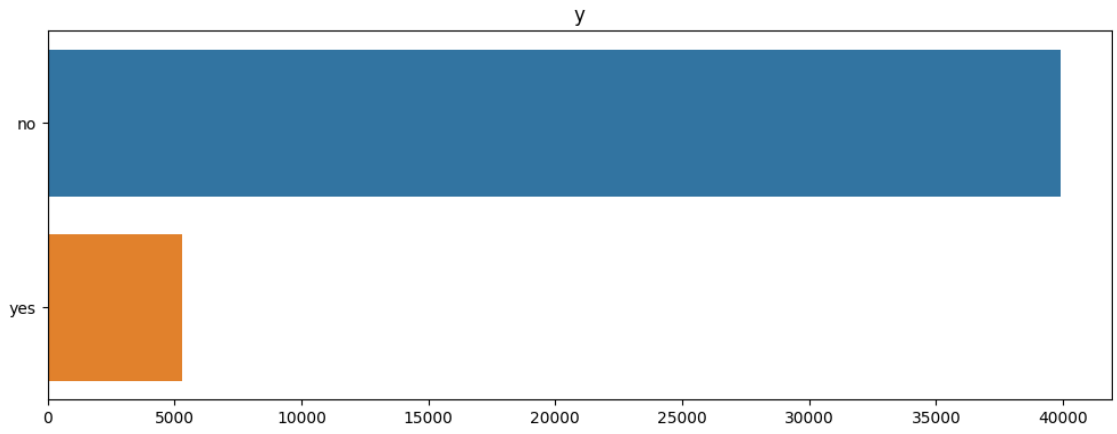

loan
no
yes
0
5000
10000
15000
20000
25000
30000
35000


contact
cellular
unknown
telephone
0
5000
10000
15000
20000
25000
30000


month
may
jul
aug
jun
nov
apr
feb
jan
oct
sep
mar
dec
0
2000
4000
6000
8000
10000
12000
14000


day
0
5
10
15
20
25
30
322 447 524 643 840 905 939 1035 1079 1121 1293 1415 1445 1479 1561 1566 1585 1603 1703 1745 1757 1817 1830 1842 1848 1910 1932 1939 2026 2308 2752



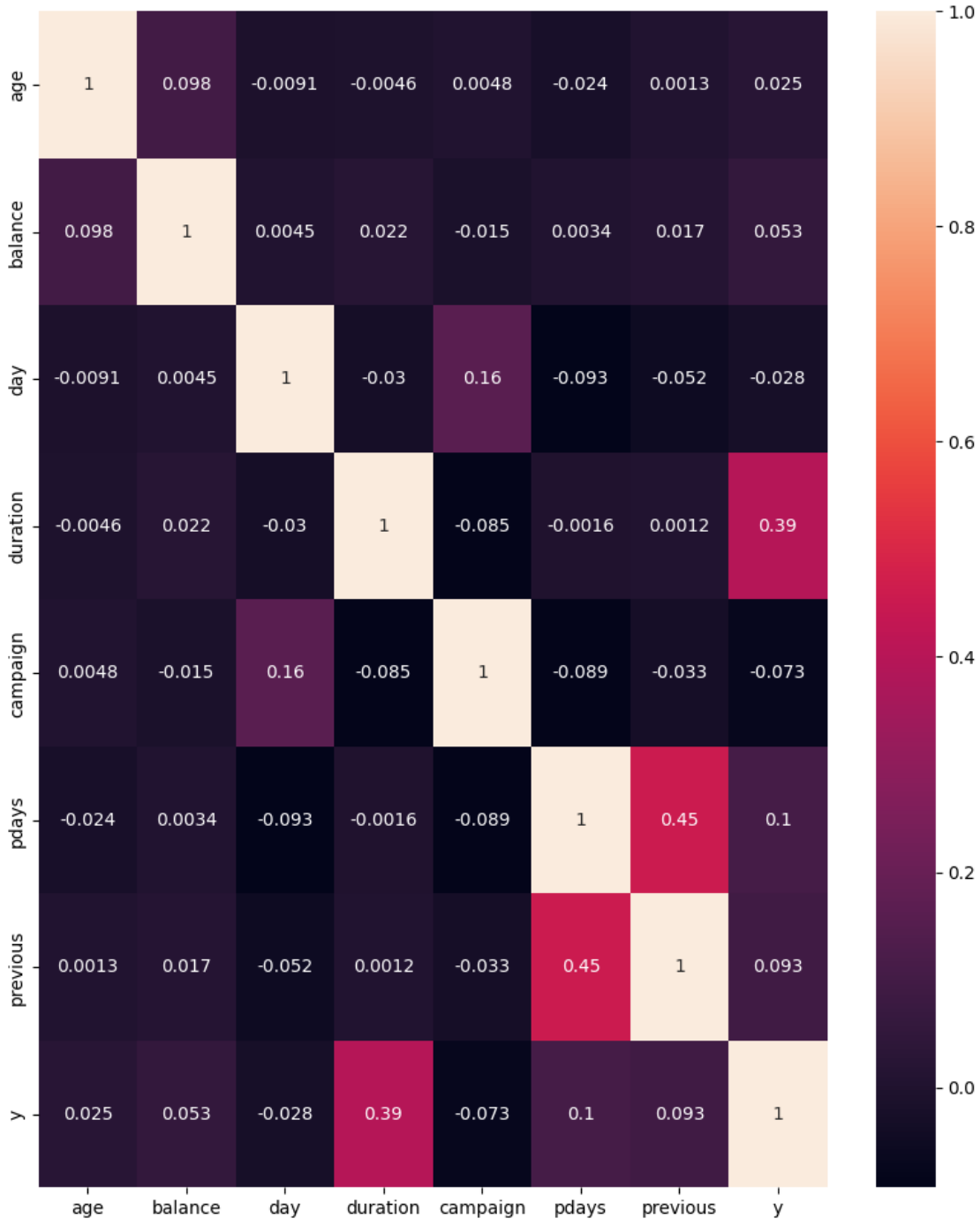


[13]:
```
df.y = df.y.map({'no':0, 'yes':1}).astype('uint8')
```

[14]:
```
import seaborn as sns
corr = df.corr()

f, ax = plt.subplots(figsize=(10,12))

sns.heatmap(corr, ax=ax, annot=True)

plt.title("Pearson correlation of Features", y=1.05, size=15)
```

```
<ipython-input-14-b583fac193a6>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr = df.corr()
```

[14]: Text(0.5, 1.05, 'Pearson correlation of Features')



[15]:
```
df.drop(['marital'],axis=1, inplace=True)
df.drop(['contact'],axis=1, inplace=True)
df.head()
```

[15]:

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

	duration	campaign	pdays	previous	poutcome	y
0	261	1	-1	0	unknown	0
1	151	1	-1	0	unknown	0
2	76	1	-1	0	unknown	0
3	92	1	-1	0	unknown	0
4	198	1	-1	0	unknown	0

[16]:
```
df[['default','housing','loan']]=df[['default','housing','loan']].
 ↪replace(["yes","no"],["1","0"])
df['month']=df['month'].replace(["jan","feb","mar","apr","may","jun","jul",␣
 ↪"aug","sep","oct","nov","dec"],["1","2","3","4","5","6","7","8","9","10","11","12"])
df['job']=df['job'].replace(['unknown'],['other'])

df.head()
```

[16]:

	age	job	education	default	balance	housing	loan	day	month	\
0	58	management	tertiary	0	2143	1	0	5	5	
1	44	technician	secondary	0	29	1	0	5	5	
2	33	entrepreneur	secondary	0	2	1	1	5	5	
3	47	blue-collar	unknown	0	1506	1	0	5	5	
4	33	other	unknown	0	1	0	0	5	5	

	duration	campaign	pdays	previous	poutcome	y
0	261	1	-1	0	unknown	0
1	151	1	-1	0	unknown	0
2	76	1	-1	0	unknown	0
3	92	1	-1	0	unknown	0
4	198	1	-1	0	unknown	0

[17]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['job']=le.fit_transform(df['job'])
df['education']=le.fit_transform(df['education'])
df['poutcome']=le.fit_transform(df['poutcome'])
```

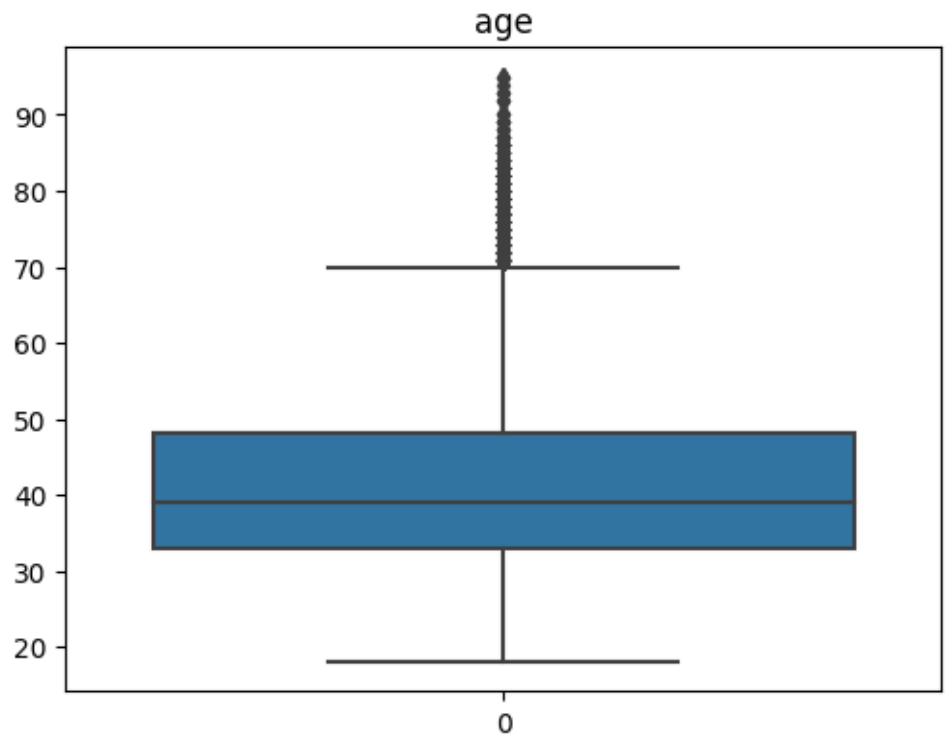
[18]:
```
df.head()
```

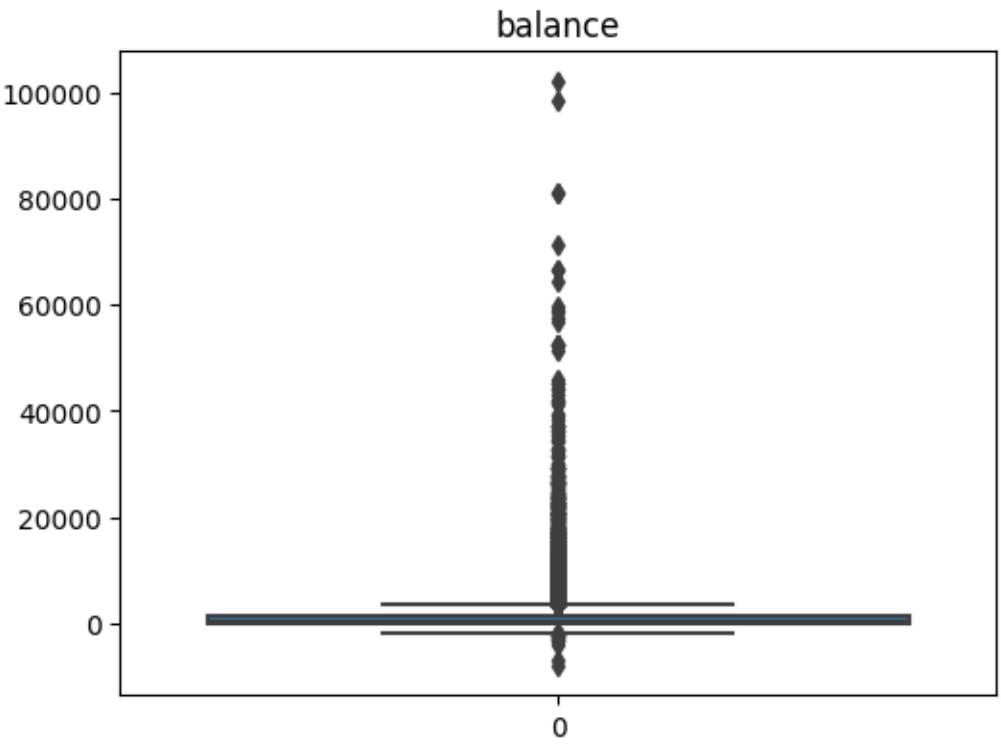
[18]:

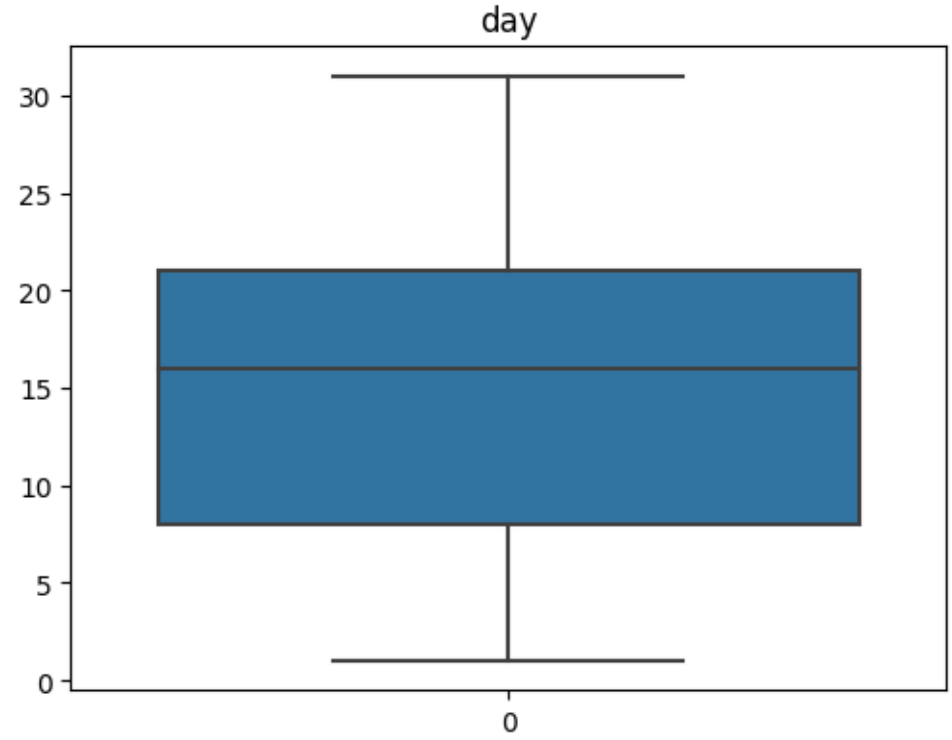
	age	job	education	default	balance	housing	loan	day	month	duration	\
0	58	4	2	0	2143	1	0	5	5	261	
1	44	10	1	0	29	1	0	5	5	151	
2	33	2	1	0	2	1	1	5	5	76	
3	47	1	3	0	1506	1	0	5	5	92	
4	33	5	3	0	1	0	0	5	5	198	

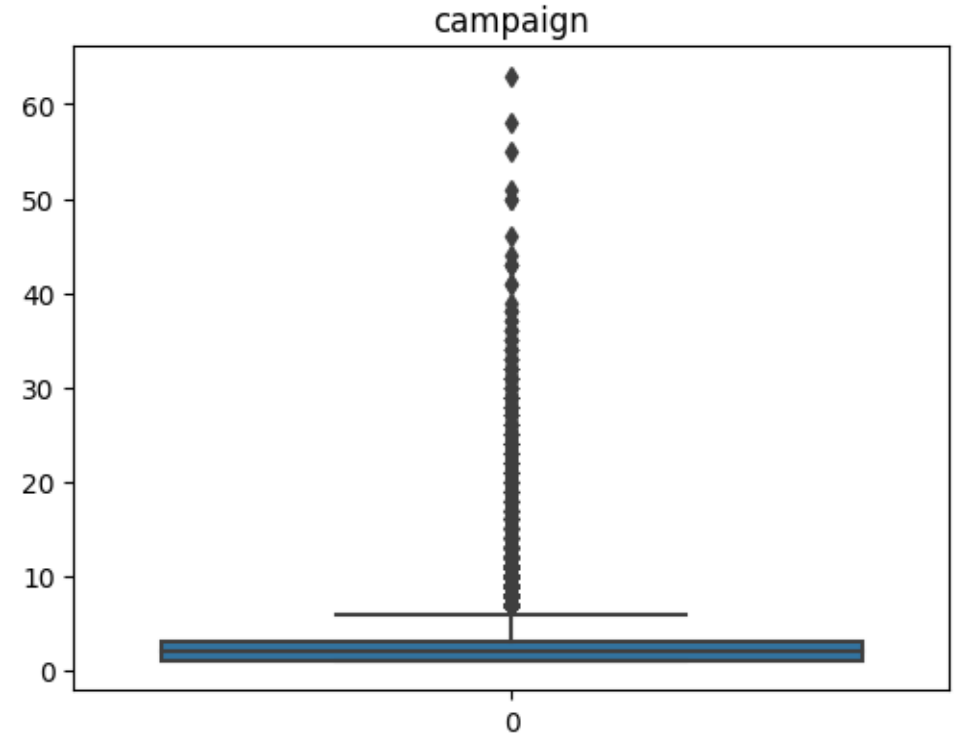
	campaign	pdays	previous	poutcome	y
0	1	-1	0	3	0
1	1	-1	0	3	0
2	1	-1	0	3	0
3	1	-1	0	3	0
4	1	-1	0	3	0

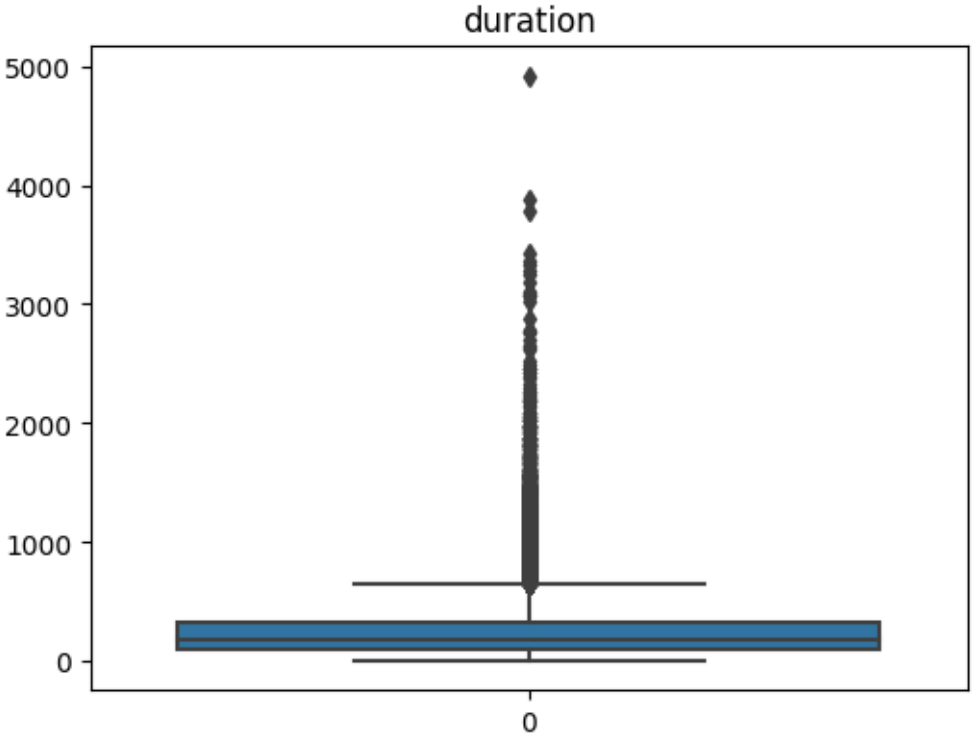
[19]:
```
col=df[['age','balance','day','campaign','duration','pdays','previous']]
for i in col:
  n=1
  plt.figure(figsize=(20,20))
  plt.subplot(4,3,1)
  sns.boxplot(df[i])
  plt.title(i)
  plt.show()
  n=n+1
```

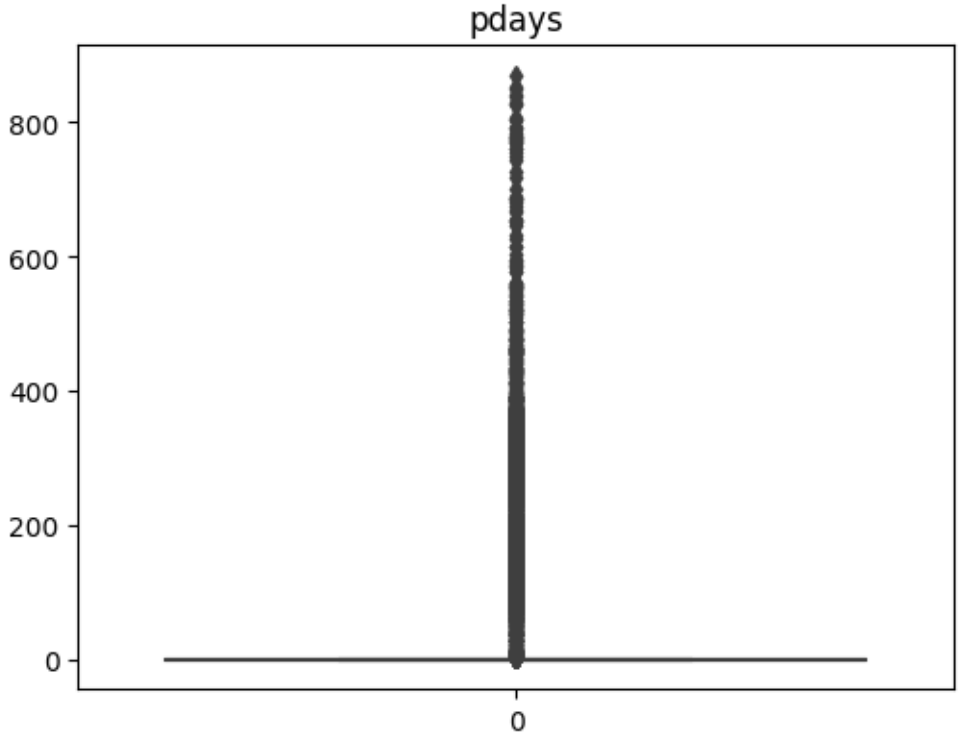


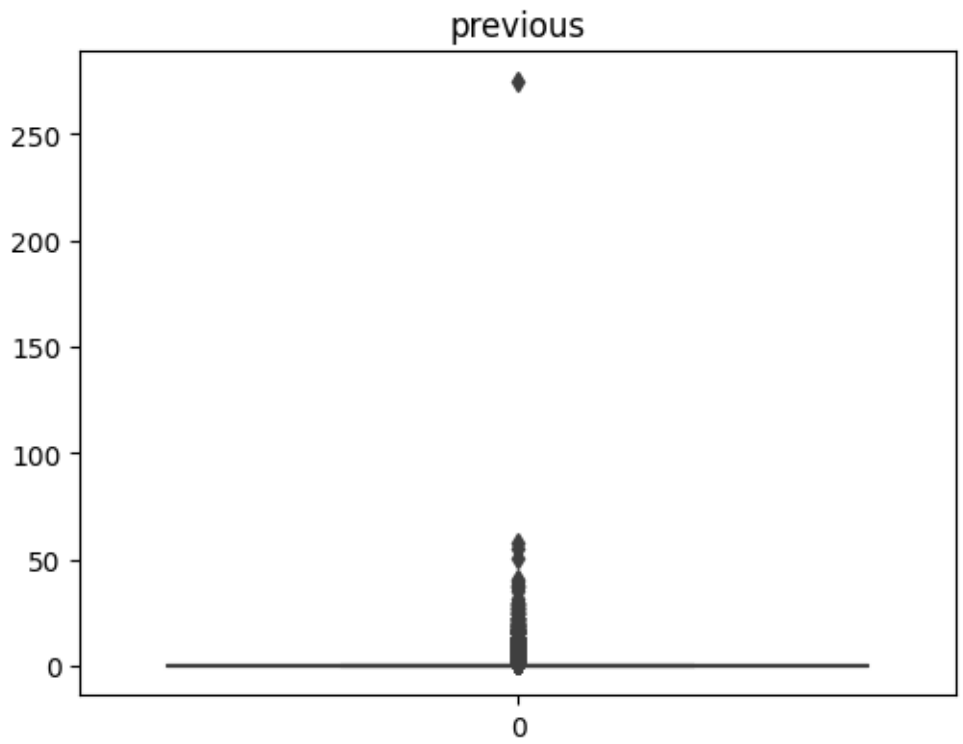

balance
100000
80000
60000
40000
20000
0
0


day
30
25
20
15
10
5
0
0


campaign
60
50
40
30
20
10
0
0


duration
5000
4000
3000
2000
1000
0
0


pdays
800
600
400
200
0
0

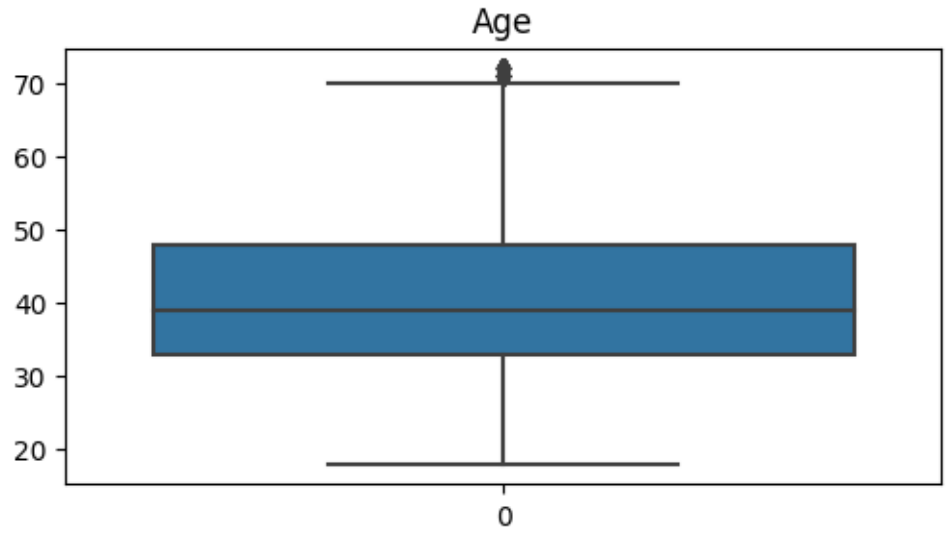


[20]:
```
from scipy import stats
import numpy as np
z = np.abs(stats.
  ↪zscore(df[['age','balance','duration','campaign','pdays','previous']]))
print(z)
df=df[(z<3).all(axis=1)]
df.shape
```

```
            age   balance  duration  campaign     pdays  previous
0      1.606965  0.256419  0.011016  0.569351  0.411453  0.251940
1      0.288529  0.437895  0.416127  0.569351  0.411453  0.251940
2      0.747384  0.446762  0.707361  0.569351  0.411453  0.251940
3      0.571051  0.047205  0.645231  0.569351  0.411453  0.251940
4      0.747384  0.447091  0.233620  0.569351  0.411453  0.251940
…           …         …         …         …         …         …
45206  0.947747  0.176460  2.791329  0.076230  0.411453  0.251940
45207  2.831227  0.120447  0.768224  0.246560  0.411453  0.251940
45208  2.925401  1.429593  3.373797  0.721811  1.436189  1.050473
45209  1.512791  0.228024  0.970146  0.399020  0.411453  0.251940
45210  0.370689  0.528364  0.399328  0.246560  1.476138  4.523577

[45211 rows x 6 columns]
```

[20]: (40209, 15)

[21]:
```
plt.figure(figsize=(20,10))
plt.subplot(3,3,1)
sns.boxplot(df['age'])
plt.title("Age")
plt.show()
```

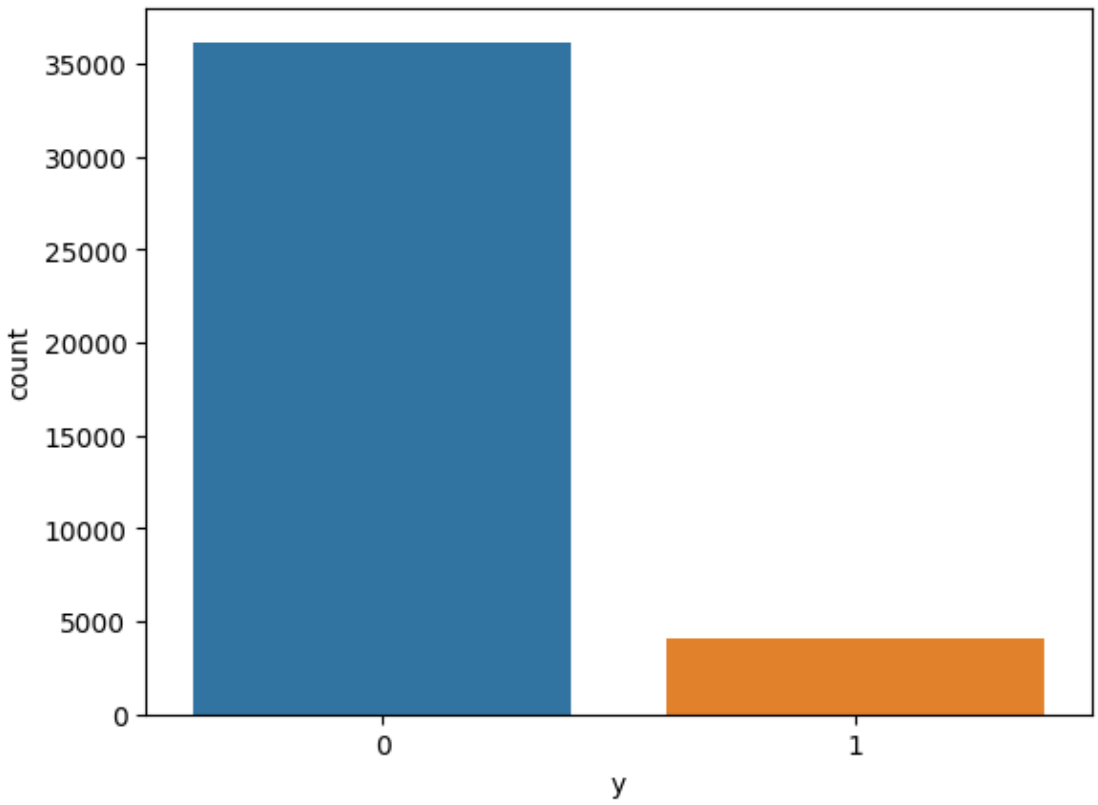


[22]:
```
x=df.drop(['y'],axis=1) #contain all  independent variable
y=df['y']               #dependent variable
df['y'].value_counts()
```

[22]:
```
0    36155
1     4054
Name: y, dtype: int64
```

[25]:
```
sns.countplot(x='y',data=df)
```

[25]:
```
<Axes: xlabel='y', ylabel='count'>
```



[ ]:

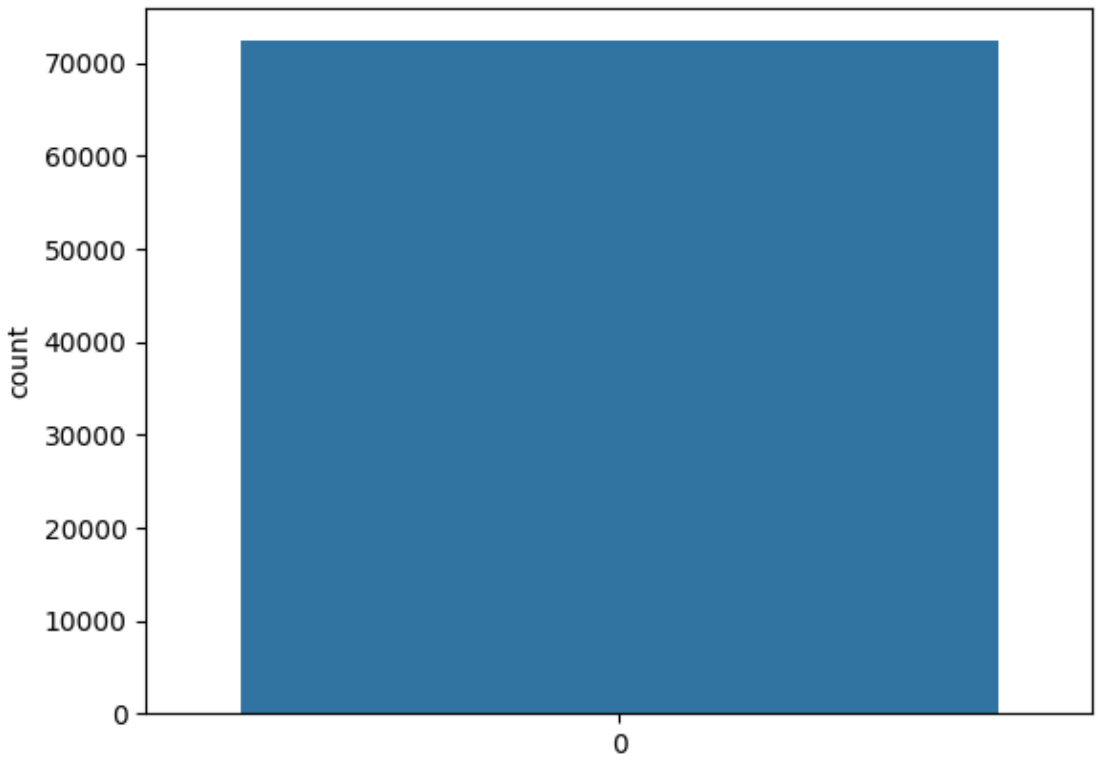
Random Over-Sampling

[33]:
```
from imblearn.over_sampling import RandomOverSampler

os =  RandomOverSampler()
x_new,y_new=os.fit_resample(x,y)
```

[34]:
```
from collections import Counter
print('Original dataset shape {}'.format(Counter(y)))
print('Resampled dataset shape {}'.format(Counter(y_new)))
sns.countplot(y_new)
```

```
Original dataset shape Counter({0: 36155, 1: 4054})
Resampled dataset shape Counter({0: 36155, 1: 36155})
```

[34]: <Axes: ylabel='count'>



[35]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import␣
 ↪accuracy_score,classification_report,confusion_matrix,roc_auc_score

#dividing the dataset into training and testing
xtrain,xtest,ytrain,ytest=train_test_split(x_new,y_new,test_size=.
 ↪20,random_state=0)
print(xtrain.shape,xtest.shape,ytrain.shape,ytest.shape)

#feature scaling
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
xtrain=scaler.fit_transform(xtrain)
xtest=scaler.transform(xtest)
```

```
(57848, 14) (14462, 14) (57848,) (14462,)
```

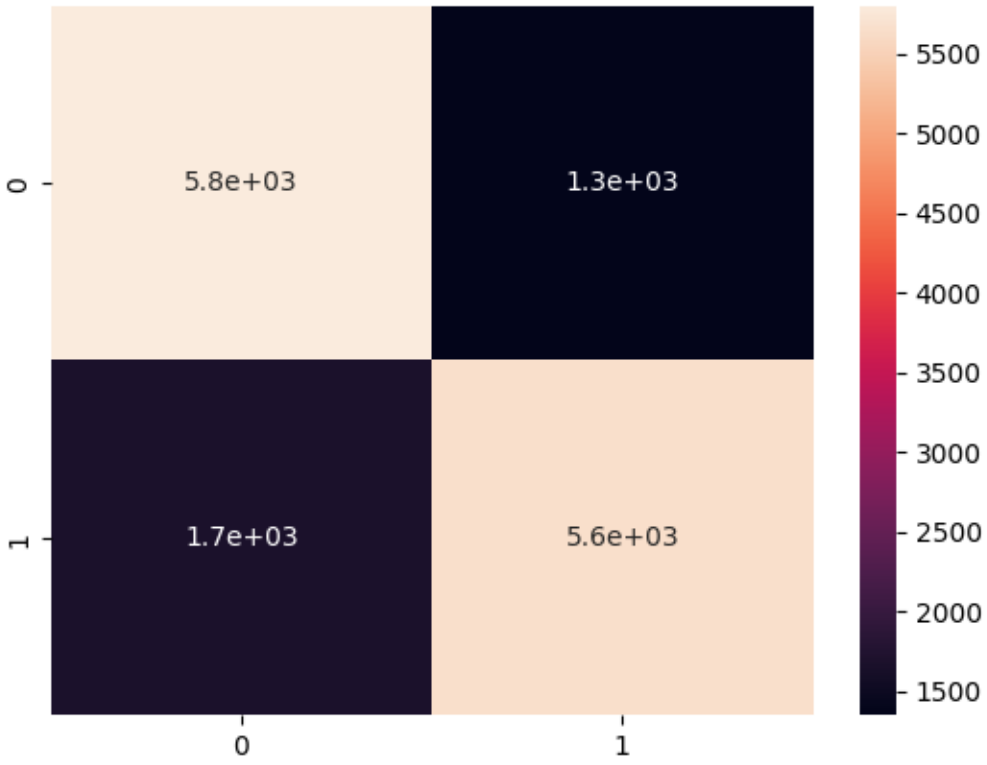
[ ]:
```

```

Logistic Regression

[36]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score,recall_score,accuracy_score,f1_score


model=LogisticRegression()
model.fit(xtrain,ytrain)
pred=model.predict(xtest)

acc_lr=accuracy_score(ytest,pred)
recall_lr=recall_score(ytest,pred)
precision_lr=precision_score(ytest,pred)
f1score_lr=f1_score(ytest,pred)
AUC_LR=roc_auc_score(pred,ytest)

print("ROC_AUC Score:",AUC_LR)

cm=confusion_matrix(ytest,pred)
print(cm)
sns.heatmap(cm,annot=True)
```

```
ROC_AUC Score: 0.7920924596219553
[[5800 1347]
 [1667 5648]]
```

[36]: <Axes: >



[37]:
```
print(classification_report(pred,ytest))
```

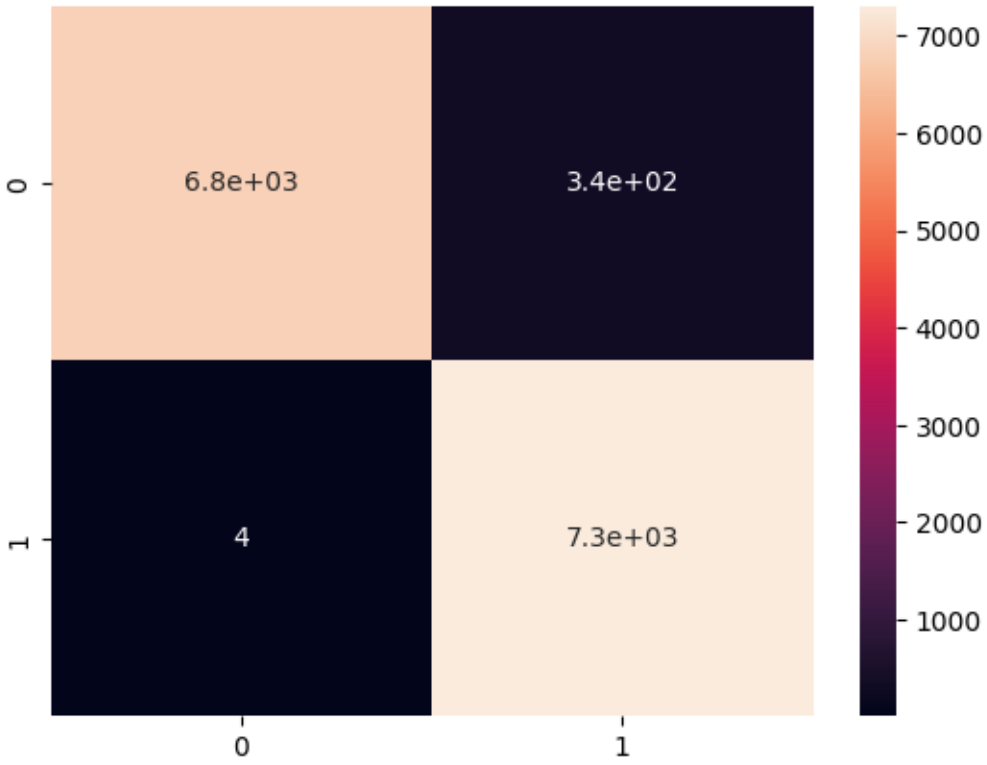
```
              precision    recall  f1-score   support

           0       0.81      0.78      0.79      7467
           1       0.77      0.81      0.79      6995

    accuracy                           0.79     14462
   macro avg       0.79      0.79      0.79     14462
weighted avg       0.79      0.79      0.79     14462
```

RandomForest Classifier

[38]:
```
from sklearn.ensemble import RandomForestClassifier

randomforest = RandomForestClassifier()
randomforest.fit(xtrain, ytrain)
y_pred = randomforest.predict(xtest)

AUC_RF=roc_auc_score(y_pred,ytest)
acc_rf=accuracy_score(ytest,y_pred)
recall_rf=recall_score(ytest,y_pred)
precision_rf=precision_score(ytest,y_pred)
f1score_rf=f1_score(ytest,y_pred)
print("ROC_AUC Score:",AUC_RF)

cm=confusion_matrix(ytest,y_pred)
print(cm)

sns.heatmap(cm,annot=True)
```

```
ROC_AUC Score: 0.9773620923583338
[[6805  342]
 [   4 7311]]
```

[38]:
```
<Axes: >
```



[39]:
```
print(classification_report(y_pred,ytest))
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.98      6809
           1       1.00      0.96      0.98      7653

    accuracy                           0.98     14462
   macro avg       0.98      0.98      0.98     14462
weighted avg       0.98      0.98      0.98     14462
```

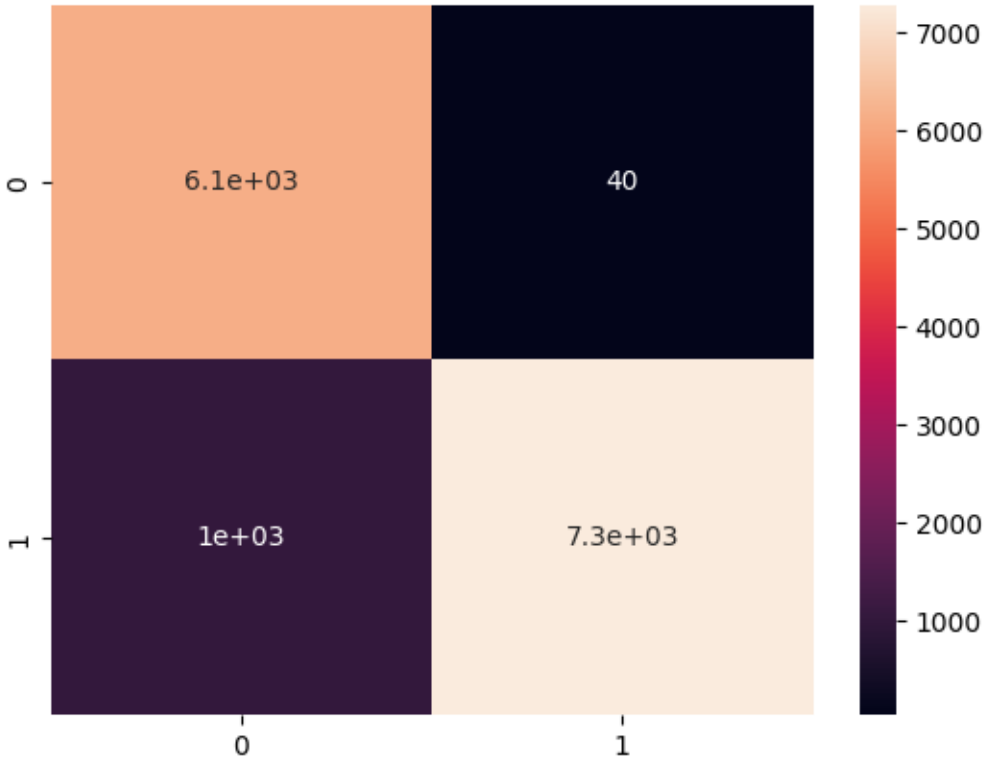
[ ]:
```
KNeighbors Classifier
```

[40]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(xtrain, ytrain)
y_predict = knn.predict(xtest)

acc_knn=accuracy_score(ytest,y_predict)
recall_knn=recall_score(ytest,y_predict)
```

```
precision_knn=precision_score(ytest,y_predict)
f1score_knn=f1_score(ytest,y_predict)

AUC_KN=roc_auc_score(y_predict,ytest)
print("ROC_AUC Score:",AUC_KN)
cm=confusion_matrix(y_predict,ytest)
print(cm)

sns.heatmap(cm,annot=True)
```

```
ROC_AUC Score: 0.9357016428345649
[[6135   40]
 [1012 7275]]
```

[40]: <Axes: >



[41]:
```
print(classification_report(y_predict,ytest))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92      6175
           1       0.99      0.88      0.93      8287
```

```
    accuracy                           0.93     14462
   macro avg       0.93      0.94      0.93     14462
weighted avg       0.94      0.93      0.93     14462
```

Best model

[42]:
```
ind=['Logistic regression','Randomforest','KNeighbors']
data={"Accuracy":[acc_lr,acc_rf,acc_knn],"Recall":
 ↪[recall_lr,recall_rf,recall_knn],"Precision":
 ↪[precision_lr,precision_rf,precision_knn],
    'f1_score':[f1score_lr,f1score_rf,f1score_knn],"ROC_AUC":
 ↪[AUC_LR,AUC_RF,AUC_KN]}
result=pd.DataFrame(data=data,index=ind)
result
```

[42]:

	Accuracy	Recall	Precision	f1_score	ROC_AUC
Logistic regression	0.791592	0.772112	0.807434	0.789378	0.792092
Randomforest	0.976075	0.999453	0.955312	0.976884	0.977362
KNeighbors	0.927258	0.994532	0.877881	0.932573	0.935702