In [86]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,roc_auc_score,accuracy_score,roc_cur
from sklearn.preprocessing import LabelEncoder

import warnings
warnings.filterwarnings('ignore')
```

In [87]:
```
bank_data = pd.read_csv('bank-full.csv',sep=';')
bank_data
```

Out[87]:

	age	job	marital	education	default	balance	housing	loan	contact	day	mor
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	m
1	44	technician	single	secondary	no	29	yes	no	unknown	5	m
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	m
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	m
4	33	unknown	single	unknown	no	1	no	no	unknown	5	m
...	...	...	...	...	...	...	...	...	...	...	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	r
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	r
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	r
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	r
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	r

45211 rows × 17 columns

In [88]:
```
bank_data.shape
```

Out[88]:
```
(45211, 17)
```

In [89]:
```
bank_data.isna().sum()
```

Out[89]:
```
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 [90]:
```
bank_data.dtypes
```

Out[90]:
```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

In [91]:
```
bank_data.describe()
```

Out[91]:

	age	balance	day	duration	campaign	pdays	p
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275

In [92]:
```
bank_data.columns
```

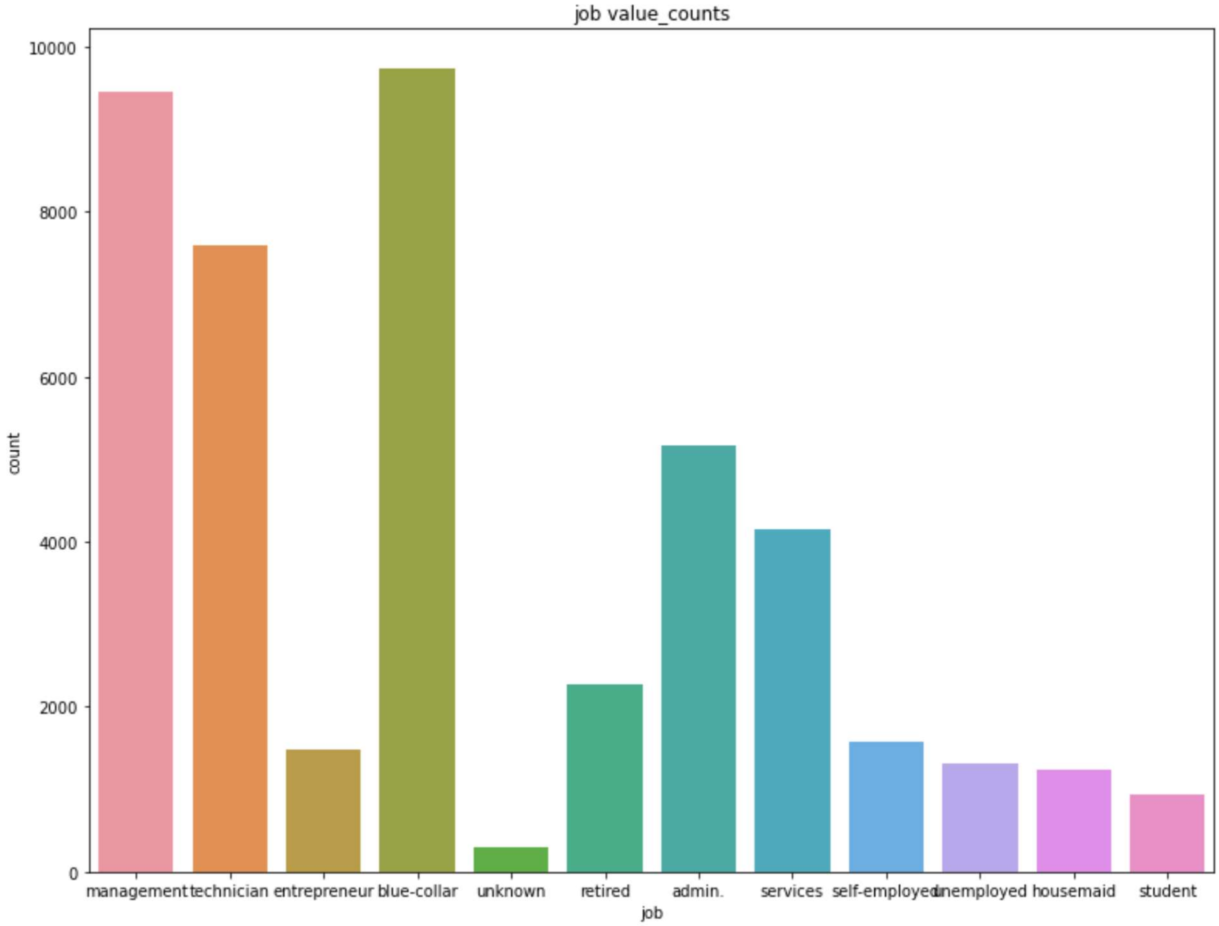
Out[92]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [93]:
```
pd.crosstab(bank_data['job'],bank_data['y'])
```

Out[93]:

y job	no	yes
admin.	4540	631
blue-collar	9024	708
entrepreneur	1364	123
housemaid	1131	109
management	8157	1301
retired	1748	516
self-employed	1392	187
services	3785	369
student	669	269
technician	6757	840
unemployed	1101	202
unknown	254	34

In [94]:
```
plt.figure(figsize=(13,10))
sns.countplot(x = bank_data['job'])
plt.title('job value_counts')
plt.show()
```

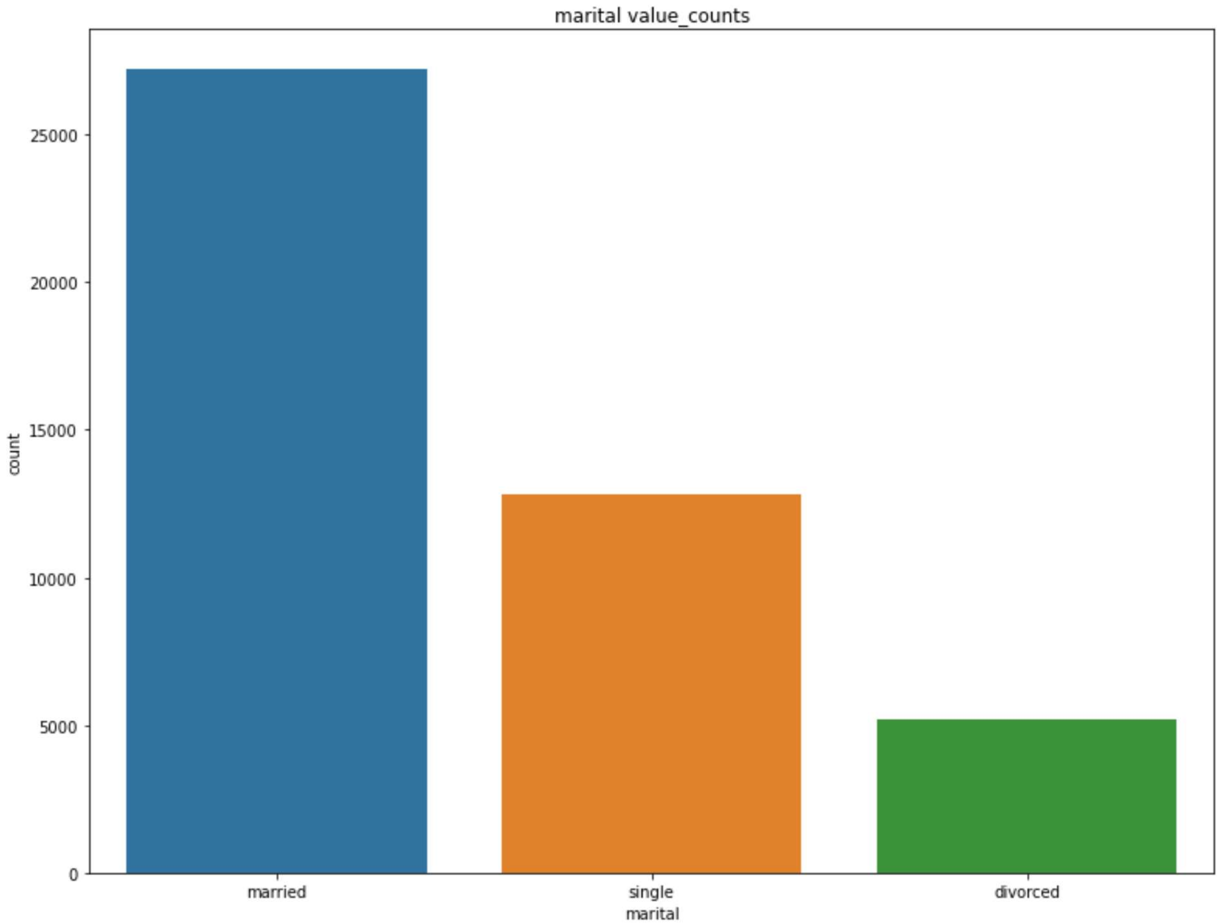


In [95]:
```
pd.crosstab(bank_data['marital'],bank_data['y'])
```

Out[95]:

y	no	yes
marital		
divorced	4585	622
married	24459	2755
single	10878	1912

In [96]:
```
plt.figure(figsize=(13,10))
sns.countplot(x = bank_data['marital'])
plt.title('marital value_counts')
plt.show()
```

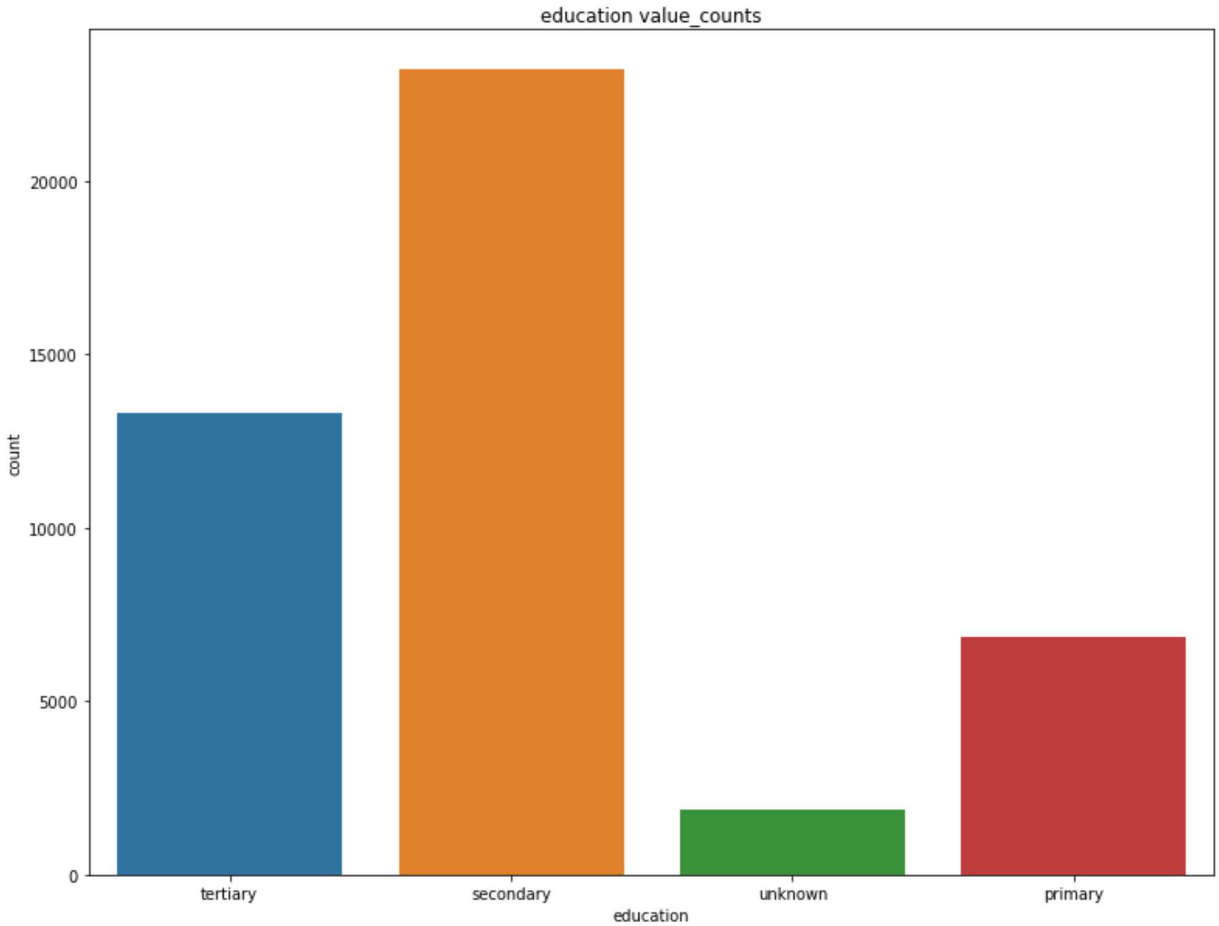


In [97]:
```
pd.crosstab(bank_data['education'],bank_data['y'])
```

Out[97]:

y	no	yes
education		
primary	6260	591
secondary	20752	2450
tertiary	11305	1996
unknown	1605	252

In [98]:
```
plt.figure(figsize=(13,10))
sns.countplot(x = bank_data['education'])
plt.title('education value_counts')
plt.show()
```

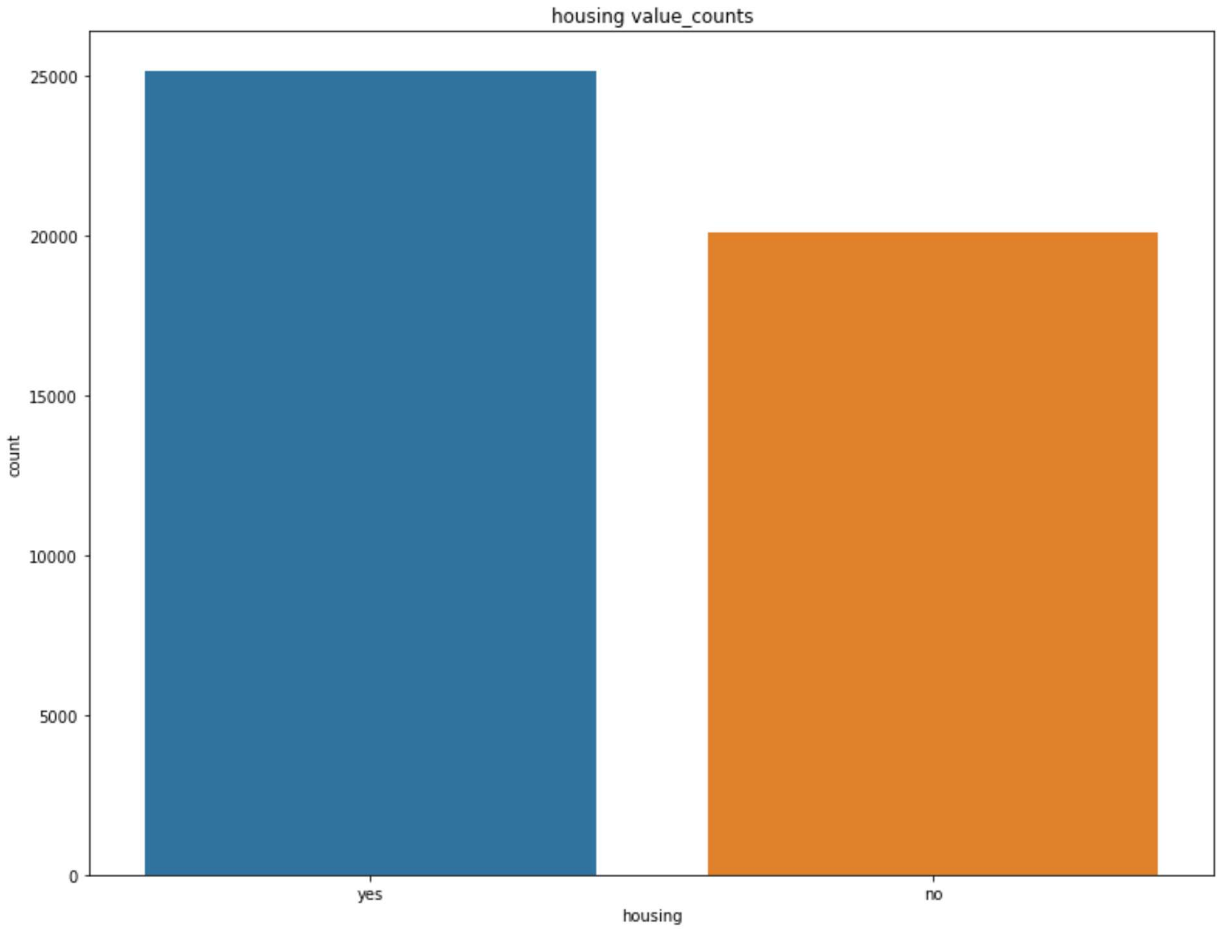


In [99]:
```
pd.crosstab(bank_data['housing'],bank_data['y'])
```

Out[99]:

y housing	no	yes
no	16727	3354
yes	23195	1935

In [100]:
```
plt.figure(figsize=(13,10))
sns.countplot(x = bank_data['housing'])
plt.title('housing value_counts')
plt.show()
```

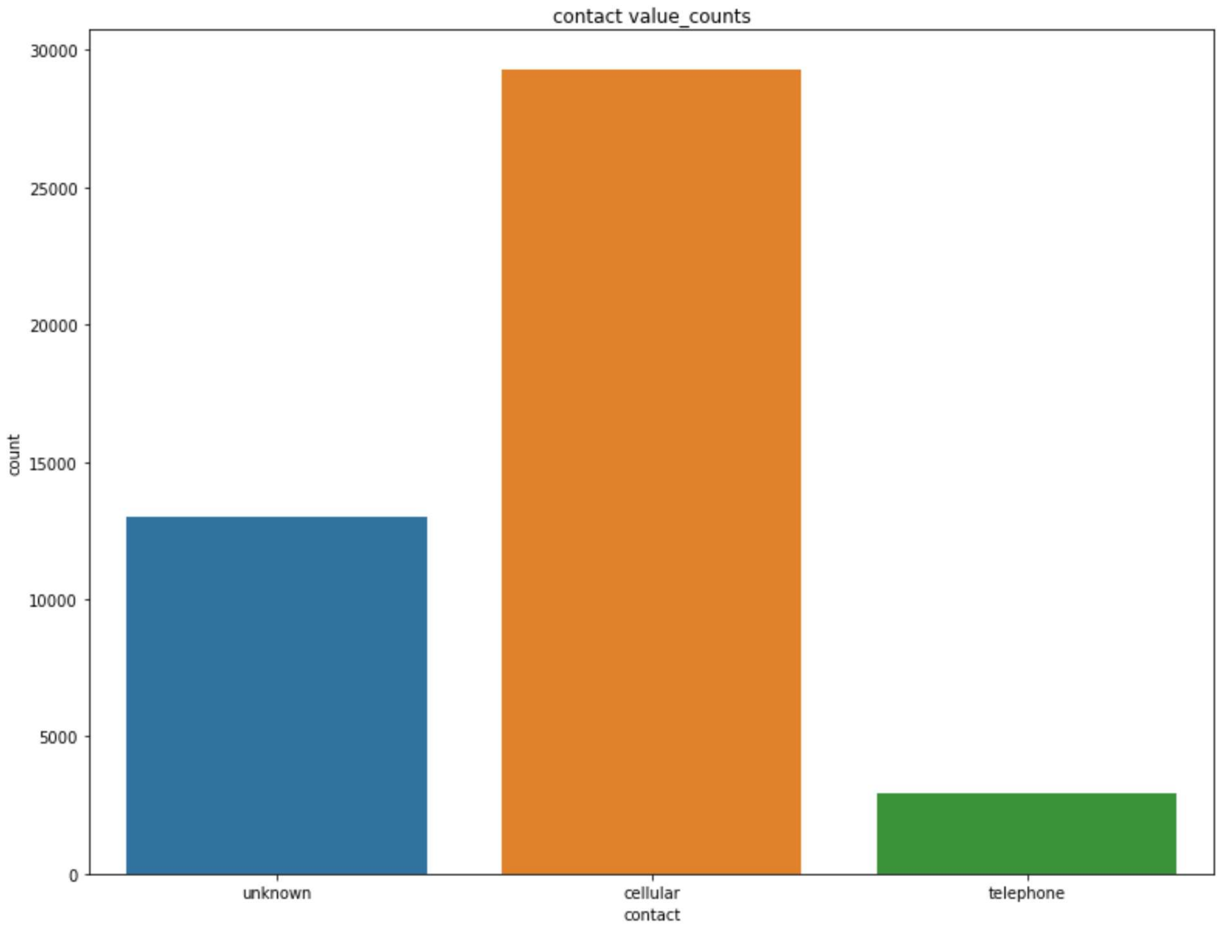


In [101]:
```
pd.crosstab(bank_data['contact'],bank_data['y'])
```

Out[101]:

y	no	yes
contact		
cellular	24916	4369
telephone	2516	390
unknown	12490	530

In [102]:
```
plt.figure(figsize=(13,10))
sns.countplot(x = bank_data['contact'])
plt.title('contact value_counts')
plt.show()
```

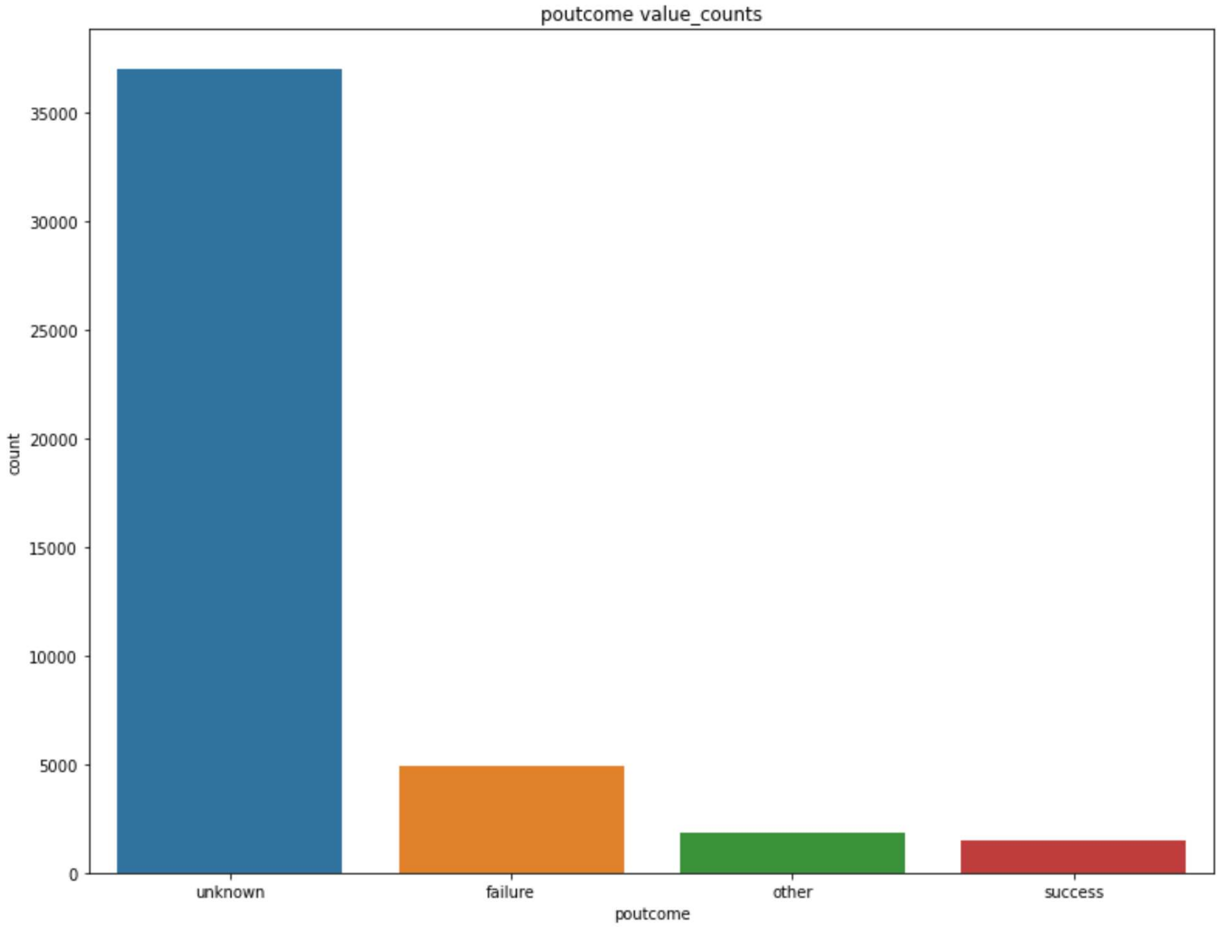


In [103]:
```
pd.crosstab(bank_data['poutcome'],bank_data['y'])
```

Out[103]:

y	no	yes
poutcome		
failure	4283	618
other	1533	307
success	533	978
unknown	33573	3386

In [104]:
```
plt.figure(figsize=(13,10))
sns.countplot(x = bank_data['poutcome'])
plt.title('poutcome value_counts')
plt.show()
```



In [105]:
```
bank_data.drop('default',axis = 1,inplace=True)
```

In [106]:
```
le = LabelEncoder()
```

In [107]:
```
bank_data['job']=le.fit_transform(bank_data['job'])
bank_data['marital']=le.fit_transform(bank_data['marital'])
bank_data['education']=le.fit_transform(bank_data['education'])
bank_data['loan']=le.fit_transform(bank_data['loan'])
bank_data['contact']=le.fit_transform(bank_data['contact'])
bank_data['y']=le.fit_transform(bank_data['y'])
bank_data['month']=le.fit_transform(bank_data['month'])
bank_data['housing']=le.fit_transform(bank_data['housing'])
bank_data['poutcome']=le.fit_transform(bank_data['poutcome'])
```

In [108]:
```
bank_data
```

Out[108]:

	age	job	marital	education	balance	housing	loan	contact	day	month	duration	campa
0	58	4	1	2	2143	1	0	2	5	8	261	
1	44	9	2	1	29	1	0	2	5	8	151	
2	33	2	1	1	2	1	1	2	5	8	76	
3	47	1	1	3	1506	1	0	2	5	8	92	
4	33	11	2	3	1	0	0	2	5	8	198	
...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	825	0	0	0	17	9	977	
45207	71	5	0	0	1729	0	0	0	17	9	456	
45208	72	5	1	1	5715	0	0	0	17	9	1127	
45209	57	1	1	1	668	0	0	1	17	9	508	
45210	37	2	1	1	2971	0	0	0	17	9	361	

45211 rows × 16 columns

In [109]:
```
x = bank_data.drop('y',axis=1)
y = bank_data[['y']]
```

In [110]:
```
std_scalar = StandardScaler()
x_scaled = std_scalar.fit_transform(x)
x_scaled = pd.DataFrame(x_scaled,columns=x.columns)
x_scaled
```

0	1.606965	-0.103820	-0.275762	1.036362	0.256419	0.893915	-0.436803	1.514306	-1
1	0.288529	1.424008	1.368372	-0.300556	-0.437895	0.893915	-0.436803	1.514306	-1
2	-0.747384	-0.714951	-0.275762	-0.300556	-0.446762	0.893915	2.289359	1.514306	-1
3	0.571051	-1.020516	-0.275762	2.373280	0.047205	0.893915	-0.436803	1.514306	-1
4	-0.747384	2.035139	1.368372	2.373280	-0.447091	-1.118674	-0.436803	1.514306	-1
...	...	...	...	...	...	...	...	...	
45206	0.947747	1.424008	-0.275762	1.036362	-0.176460	-1.118674	-0.436803	-0.713012	C
45207	2.831227	0.201746	-1.919895	-1.637474	0.120447	-1.118674	-0.436803	-0.713012	C
45208	2.925401	0.201746	-0.275762	-0.300556	1.429593	-1.118674	-0.436803	-0.713012	C
45209	1.512791	-1.020516	-0.275762	-0.300556	-0.228024	-1.118674	-0.436803	0.400647	C
45210	-0.370689	-0.714951	-0.275762	-0.300556	0.528364	-1.118674	-0.436803	-0.713012	C

45211 rows × 15 columns

In [111]:
```
linear_model = LogisticRegression()
```

In [112]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=
```

In [113]:
```
x_train.shape,y_train.shape
```

Out[113]: `((36168, 15), (36168, 1))`

In [114]:
```
x_test.shape,y_test.shape
```

Out[114]: `((9043, 15), (9043, 1))`

In [115]:
```
linear_model.fit(x_train,y_train)
```

Out[115]: `LogisticRegression()`

In [116]:
```
y_pred_train = linear_model.predict(x_train)
```

In [117]:
```
print('Accuracy_score :',accuracy_score(y_train,y_pred_train))
```

```
Accuracy_score : 0.887082503870825
```

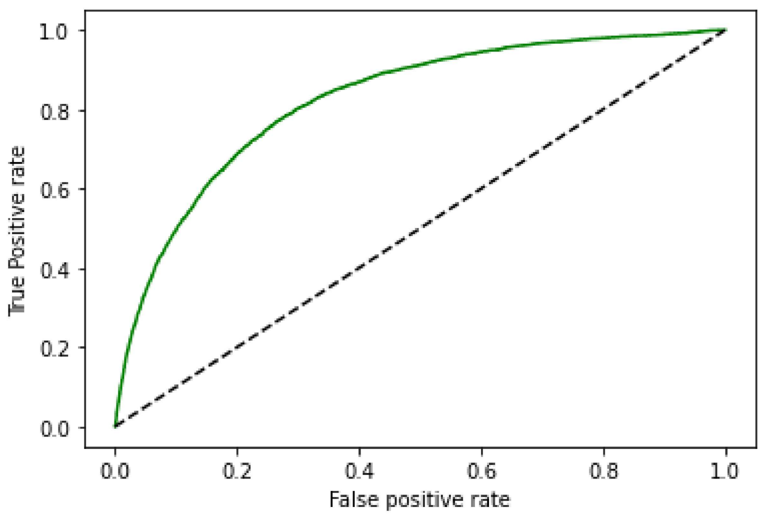
In [118]:
```
print('Confusion matrix :\n',confusion_matrix(y_train,y_pred_train))
```

```
Confusion matrix :
 [[31403   526]
 [ 3558   681]]
```

In [119]:
```
 auc =roc_auc_score(y_train,y_pred_train)
print('Auc:' ,auc)
```

```
Auc: 0.5720885225771963
```

In [120]:
```
fpr , tpr,thresholds = roc_curve(y_train,linear_model.predict_proba(x_train)[:,1]
plt.plot(fpr,tpr,color = 'green',label='logit model (area = %0.2f)'%auc)
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False positive rate')
plt.ylabel('True Positive rate')
plt.show()
```



In [121]:
```
y_pred_test = linear_model.predict(x_test)
```

In [122]:
```
print('Accuracy_score :',accuracy_score(y_test,y_pred_test))

print('Confusion matrix :\n',confusion_matrix(y_test,y_pred_test))
```

```
Accuracy_score : 0.8900807254229791
Confusion matrix :
 [[7859  134]
 [ 860  190]]
```

In [123]:
```
auc =roc_auc_score(y_test,y_pred_test)
print('Auc:' ,auc)
```

```
Auc: 0.5820938559334657
```

In [124]:
```
fpr,tpr,thresholds = roc_curve(y_test,linear_model.predict_proba(x_test)[:,1])
plt.plot(fpr,tpr,color ='red')
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
```

Out[124]:
```
Text(0, 0.5, 'True positive rate')
```