In [1]:

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

In [2]:

```
#Load the dataset
df = pd.read_csv("bank-additional-full.csv",sep=';')
```

In [3]:

```
df.head().T
```

Out[3]:

	0	1	2	3	4
age	56	57	37	40	56
job	housemaid	services	services	admin.	services
marital	married	married	married	married	married
education	basic.4y	high.school	high.school	basic.6y	high.school
default	no	unknown	no	no	no
housing	no	no	yes	no	no
loan	no	no	no	no	yes
contact	telephone	telephone	telephone	telephone	telephone
month	may	may	may	may	may
day_of_week	mon	mon	mon	mon	mon
duration	261	149	226	151	307
campaign	1	1	1	1	1
pdays	999	999	999	999	999
previous	0	0	0	0	0
poutcome	nonexistent	nonexistent	nonexistent	nonexistent	nonexistent
emp.var.rate	1.1	1.1	1.1	1.1	1.1
cons.price.idx	93.994	93.994	93.994	93.994	93.994
cons.conf.idx	-36.4	-36.4	-36.4	-36.4	-36.4
euribor3m	4.857	4.857	4.857	4.857	4.857
nr.employed	5191	5191	5191	5191	5191
y	no	no	no	no	no

In [4]:

```
# Finding the name of the columns
df.columns
```

Out[4]:

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

In [5]:

```
# Finding the data types of the column
df.dtypes
```

Out[5]:

```
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
```

In [6]:

```
#Finding number of rows and columns
df.shape
```

Out[6]:

```
(41188, 21)
```

In [7]:

```
for i in df.columns:
  print(i)
  print(df[i].unique())
  print('---'*10)
```

```
age
[56 57 37 40 45 59 41 24 25 29 35 54 46 50 39 30 55 49 34 52 58 32 38 44
 42 60 53 47 51 48 33 31 43 36 28 27 26 22 23 20 21 61 19 18 70 66 76 67
 73 88 95 77 68 75 63 80 62 65 72 82 64 71 69 78 85 79 83 81 74 17 87 91
 86 98 94 84 92 89]
------------------------------
job
['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']
------------------------------
marital
['married' 'single' 'divorced' 'unknown']
------------------------------
education
['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course'
 'unknown' 'university.degree' 'illiterate']
------------------------------
default
['no' 'unknown' 'yes']
------------------------------
housing
['no' 'yes' 'unknown']
------------------------------
loan
['no' 'yes' 'unknown']
------------------------------
contact
['telephone' 'cellular']
------------------------------
month
['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'apr' 'sep']
------------------------------
day_of_week
['mon' 'tue' 'wed' 'thu' 'fri']
------------------------------
duration
[ 261  149  226 ... 1246 1556 1868]
------------------------------
campaign
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 19 18 23 14 22 25 16 17 15 20 56
 39 35 42 28 26 27 32 21 24 29 31 30 41 37 40 33 34 43]
------------------------------
pdays
[999   6   4   3   5   1   0  10   7   8   9  11   2  12  13  14  15  16
  21  17  18  22  25  26  19  27  20]
------------------------------
previous
[0 1 2 3 4 5 6 7]
------------------------------
poutcome
['nonexistent' 'failure' 'success']
------------------------------
emp.var.rate
```

```
[ 1.1  1.4 -0.1 -0.2 -1.8 -2.9 -3.4 -3.  -1.7 -1.1]
------------------------------
cons.price.idx
[93.994 94.465 93.918 93.444 93.798 93.2   92.756 92.843 93.075 92.893
 92.963 92.469 92.201 92.379 92.431 92.649 92.713 93.369 93.749 93.876
 94.055 94.215 94.027 94.199 94.601 94.767]
------------------------------
cons.conf.idx
[-36.4 -41.8 -42.7 -36.1 -40.4 -42.  -45.9 -50.  -47.1 -46.2 -40.8 -33.6
 -31.4 -29.8 -26.9 -30.1 -33.  -34.8 -34.6 -40.  -39.8 -40.3 -38.3 -37.5
 -49.5 -50.8]
------------------------------
euribor3m
[4.857 4.856 4.855 4.859 4.86  4.858 4.864 4.865 4.866 4.967 4.961 4.959
 4.958 4.96  4.962 4.955 4.947 4.956 4.966 4.963 4.957 4.968 4.97  4.965
 4.964 5.045 5.    4.936 4.921 4.918 4.912 4.827 4.794 4.76  4.733 4.7
 4.663 4.592 4.474 4.406 4.343 4.286 4.245 4.223 4.191 4.153 4.12  4.076
 4.021 3.901 3.879 3.853 3.816 3.743 3.669 3.563 3.488 3.428 3.329 3.282
 3.053 1.811 1.799 1.778 1.757 1.726 1.703 1.687 1.663 1.65  1.64  1.629
 1.614 1.602 1.584 1.574 1.56  1.556 1.548 1.538 1.531 1.52  1.51  1.498
 1.483 1.479 1.466 1.453 1.445 1.435 1.423 1.415 1.41  1.405 1.406 1.4
 1.392 1.384 1.372 1.365 1.354 1.344 1.334 1.327 1.313 1.299 1.291 1.281
 1.266 1.25  1.244 1.259 1.264 1.27  1.262 1.26  1.268 1.286 1.252 1.235
 1.224 1.215 1.206 1.099 1.085 1.072 1.059 1.048 1.044 1.029 1.018 1.007
 0.996 0.979 0.969 0.944 0.937 0.933 0.927 0.921 0.914 0.908 0.903 0.899
 0.884 0.883 0.881 0.879 0.873 0.869 0.861 0.859 0.854 0.851 0.849 0.843
 0.838 0.834 0.829 0.825 0.821 0.819 0.813 0.809 0.803 0.797 0.788 0.781
 0.778 0.773 0.771 0.77  0.768 0.766 0.762 0.755 0.749 0.743 0.741 0.739
 0.75  0.753 0.754 0.752 0.744 0.74  0.742 0.737 0.735 0.733 0.73  0.731
 0.728 0.724 0.722 0.72  0.719 0.716 0.715 0.714 0.718 0.721 0.717 0.712
 0.71  0.709 0.708 0.706 0.707 0.7   0.655 0.654 0.653 0.652 0.651 0.65
 0.649 0.646 0.644 0.643 0.639 0.637 0.635 0.636 0.634 0.638 0.64  0.642
 0.645 0.659 0.663 0.668 0.672 0.677 0.682 0.683 0.684 0.685 0.688 0.69
 0.692 0.695 0.697 0.699 0.701 0.702 0.704 0.711 0.713 0.723 0.727 0.729
 0.732 0.748 0.761 0.767 0.782 0.79  0.793 0.802 0.81  0.822 0.827 0.835
 0.84  0.846 0.87  0.876 0.885 0.889 0.893 0.896 0.898 0.9   0.904 0.905
 0.895 0.894 0.891 0.89  0.888 0.886 0.882 0.88  0.878 0.877 0.942 0.953
 0.956 0.959 0.965 0.972 0.977 0.982 0.985 0.987 0.993 1.    1.008 1.016
 1.025 1.032 1.037 1.043 1.045 1.047 1.05  1.049 1.046 1.041 1.04  1.039
 1.035 1.03  1.031 1.028]
------------------------------
nr.employed
[5191.  5228.1 5195.8 5176.3 5099.1 5076.2 5017.5 5023.5 5008.7 4991.6
 4963.6]
------------------------------
y
['no' 'yes']
------------------------------
```

In [8]:

```
# cat stands for categorical value
cat = df[['job','marital','education','default', 'housing', 'loan',
       'contact', 'month', 'day_of_week','poutcome','y']]

# num stands for numerical variable
num = df[['age','duration', 'campaign', 'pdays',
       'previous','emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed']]
```

In [9]:

```
### numerical
numerical_cols = list(df.select_dtypes(exclude=['object']))
numerical_cols
```

Out[9]:

```
['age',
 'duration',
 'campaign',
 'pdays',
 'previous',
 'emp.var.rate',
 'cons.price.idx',
 'cons.conf.idx',
 'euribor3m',
 'nr.employed']
```

In [10]:

```
### categorical
category_cols = list(df.select_dtypes(include=['object']))
category_cols
```

Out[10]:

```
['job',
 'marital',
 'education',
 'default',
 'housing',
 'loan',
 'contact',
 'month',
 'day_of_week',
 'poutcome',
 'y']
```

In [11]:

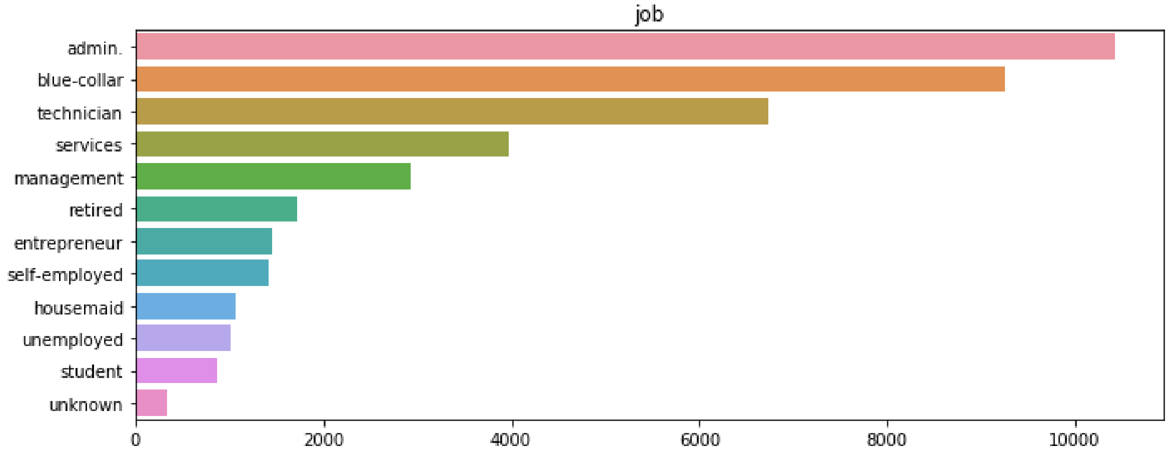
```
# Finding Missing Value
df.isnull().sum()
# There is no missing value in the data set
```

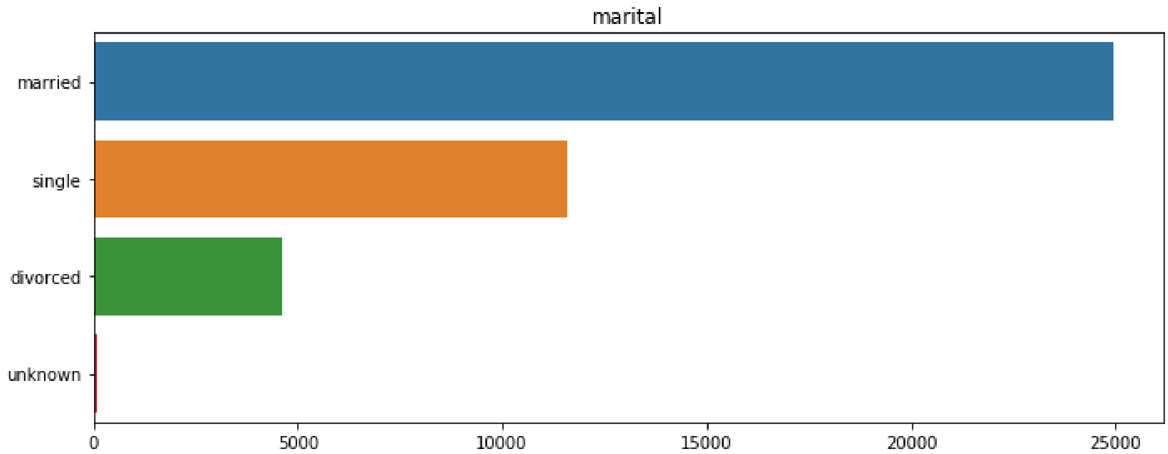
Out[11]:

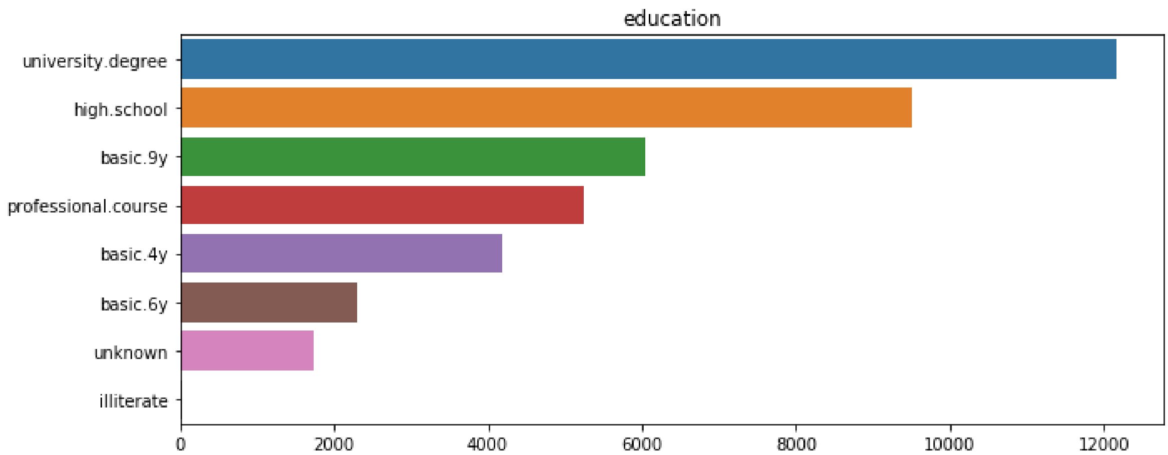
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

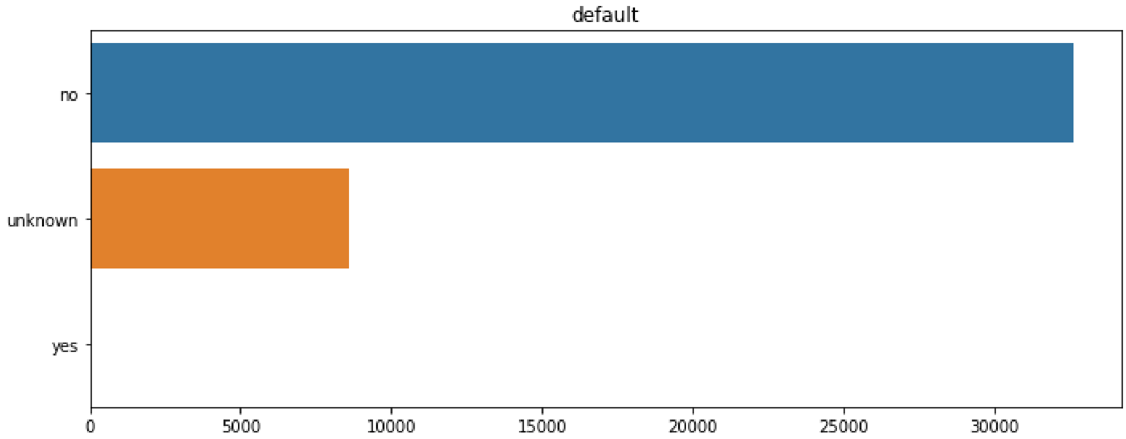
In [12]:

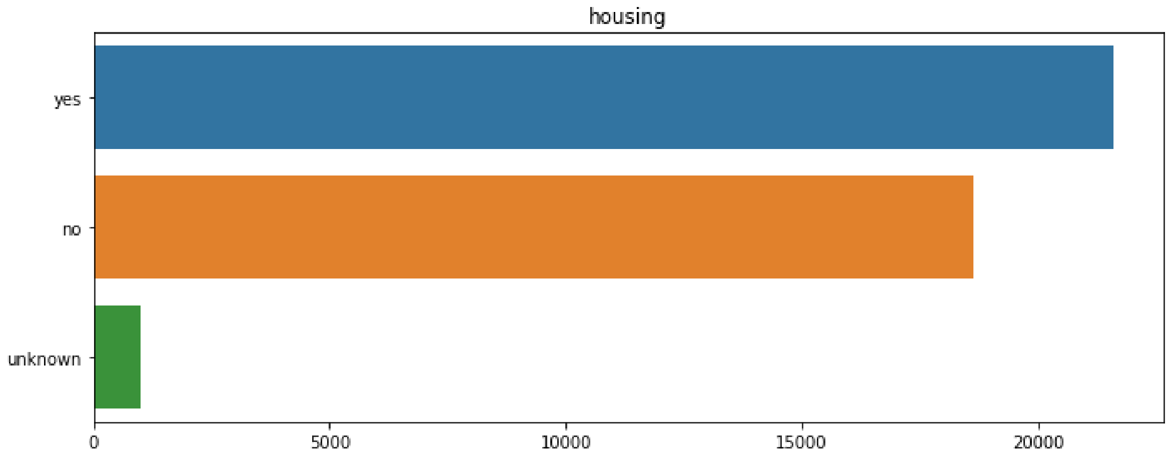
```
# Various Bar Graph of Categorical Variable
for col in category_cols:
    plt.figure(figsize=(10,4))
    sns.barplot(df[col].value_counts().values, df[col].value_counts().index)
    plt.title(col)
    plt.tight_layout()
```

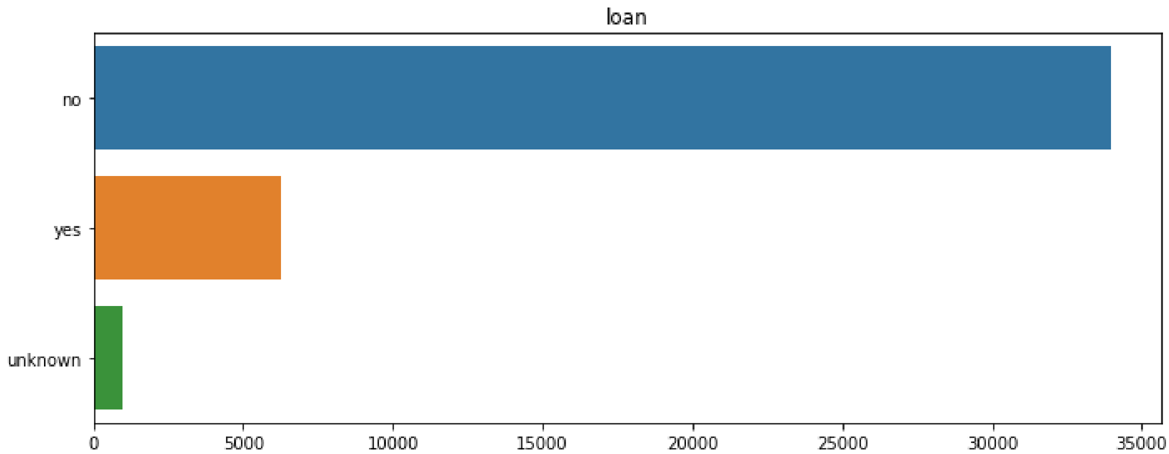


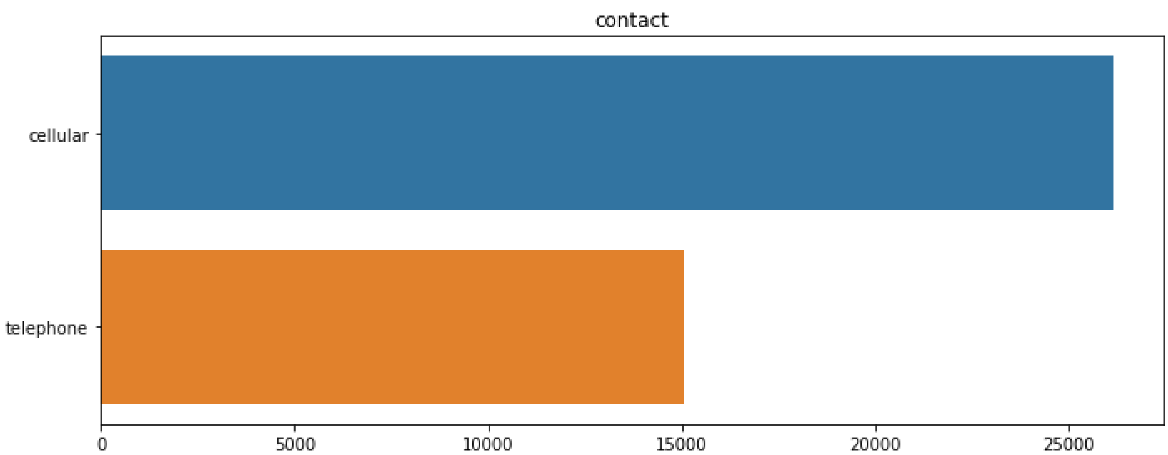


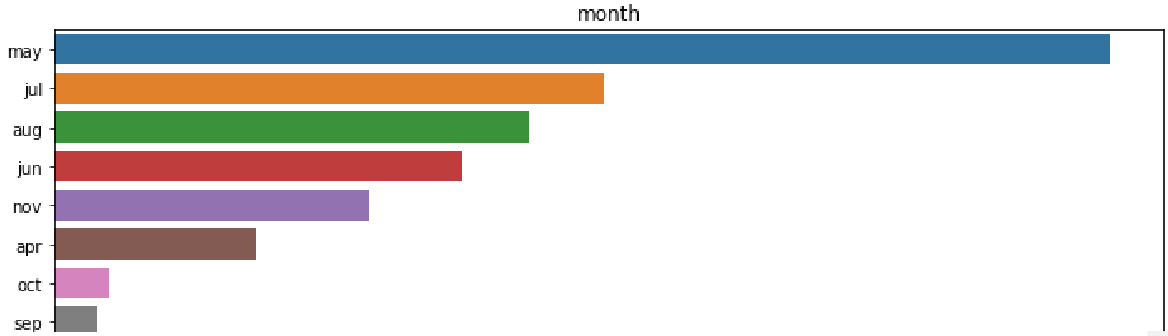


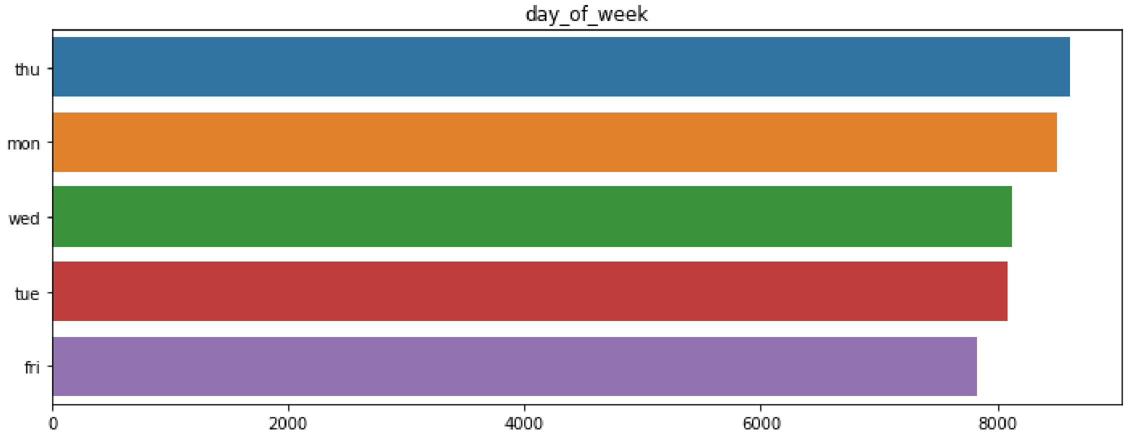

default
no
unknown
yes
0
5000
10000
15000
20000
25000
30000

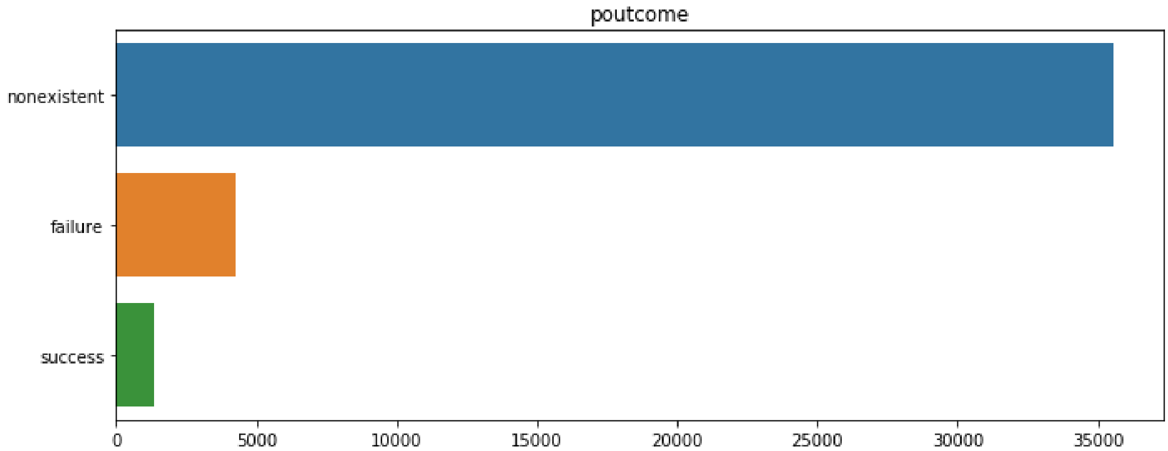

housing
yes
no
unknown
0
5000
10000
15000
20000

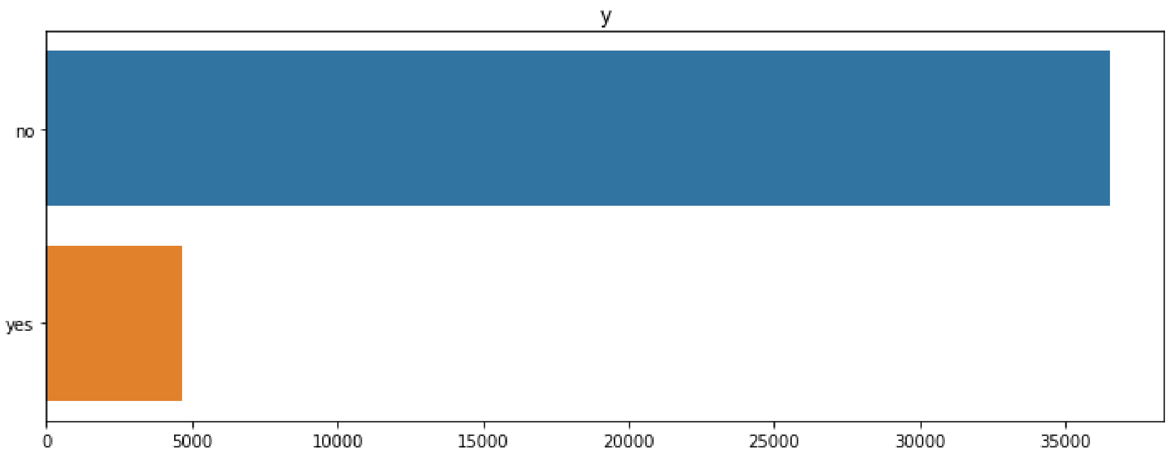

loan
no
yes
unknown
0
5000
10000
15000
20000
25000
30000
35000


contact
cellular
telephone
0
5000
10000
15000
20000
25000


month
may
jul
aug
jun
nov
apr
oct
sep

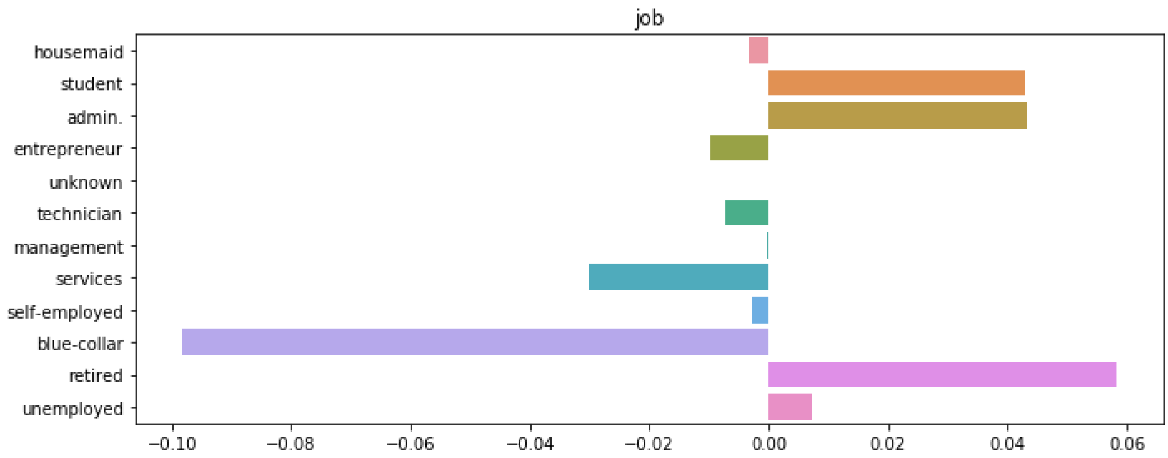

day_of_week
thu
mon
wed
tue
fri
0
2000
4000
6000
8000

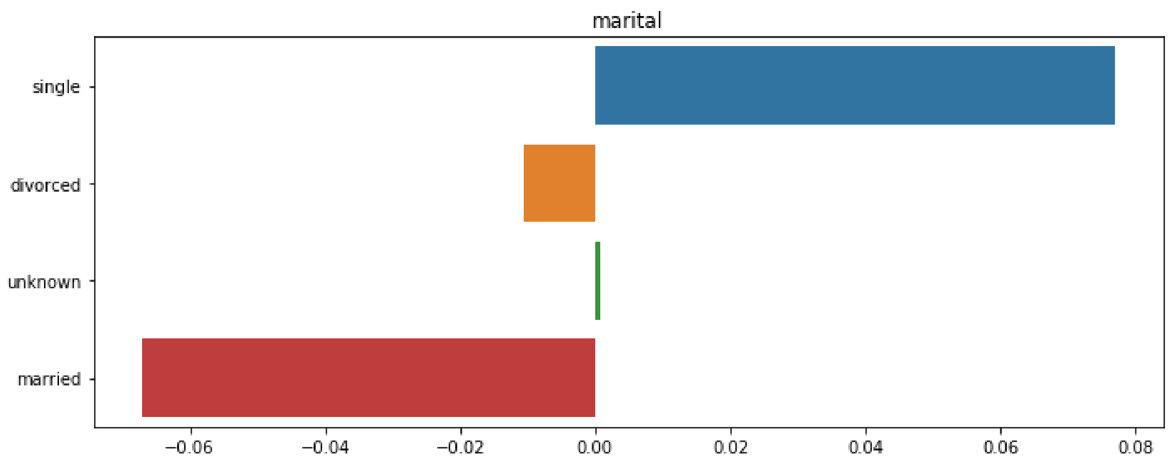

poutcome
nonexistent
failure
success
0
5000
10000
15000
20000
25000
30000
35000

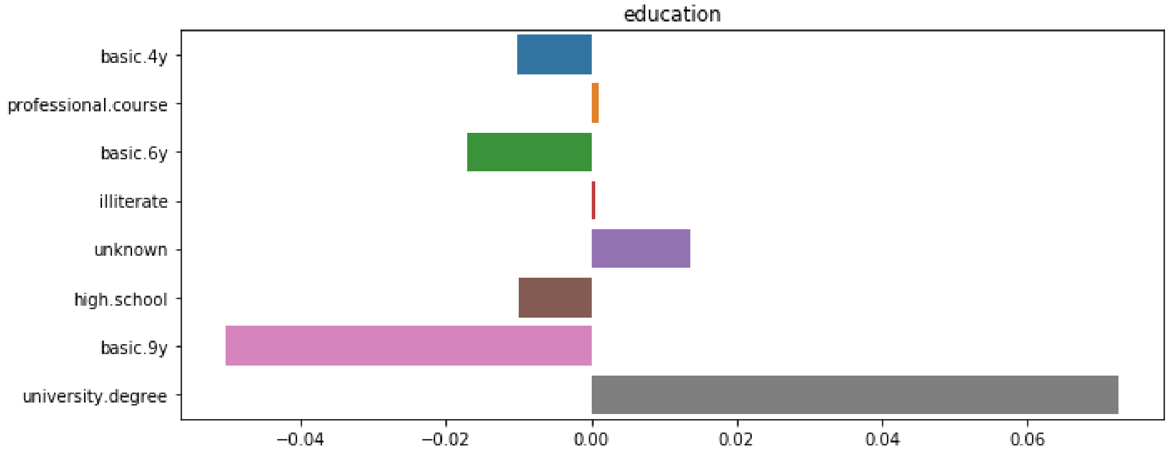

y
no
yes
0
5000
10000
15000
20000
25000
30000
35000

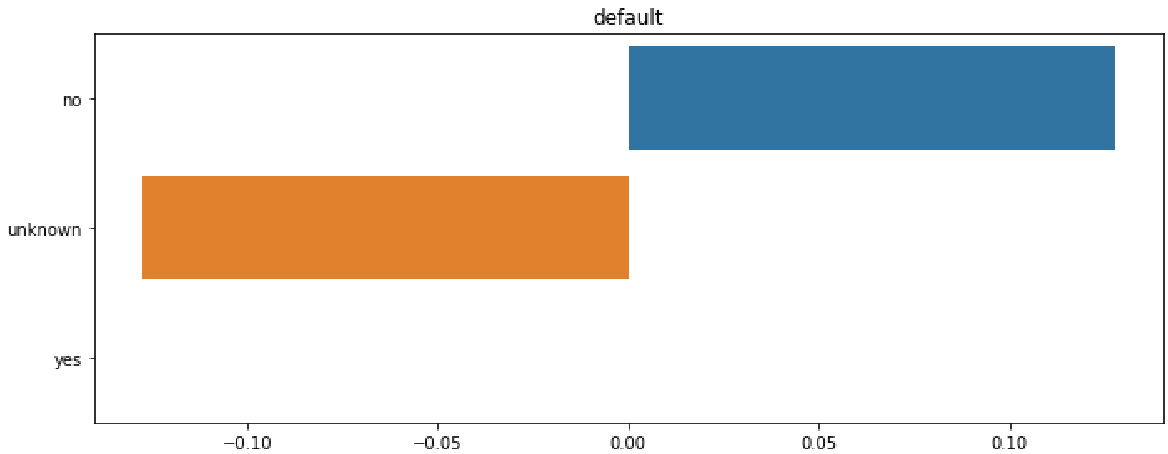
In [13]:

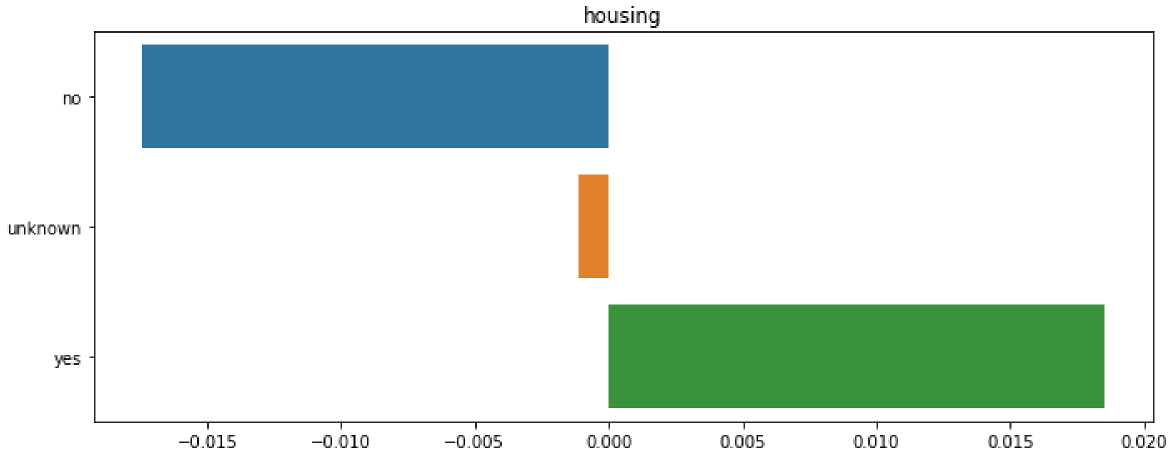
```
for col in category_cols:
    plt.figure(figsize=(10,4))
    #Returns counts of unique values for each outcome for each feature.
    pos_counts = df.loc[df.y.values == 'yes', col].value_counts()
    neg_counts = df.loc[df.y.values == 'no', col].value_counts()

    all_counts = list(set(list(pos_counts.index) + list(neg_counts.index)))

    #Counts of how often each outcome was recorded.
    freq_pos = (df.y.values == 'yes').sum()
    freq_neg = (df.y.values == 'no').sum()

    pos_counts = pos_counts.to_dict()
    neg_counts = neg_counts.to_dict()

    all_index = list(all_counts)
    all_counts = [pos_counts.get(k, 0) / freq_pos - neg_counts.get(k, 0) / freq_neg for k i

    sns.barplot(all_counts, all_index)
    plt.title(col)
    plt.tight_layout()
```

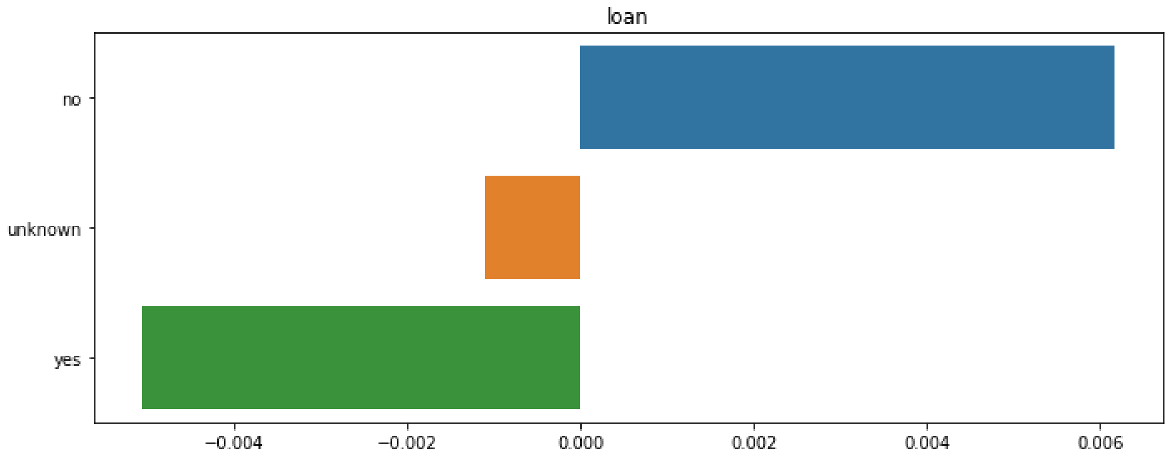


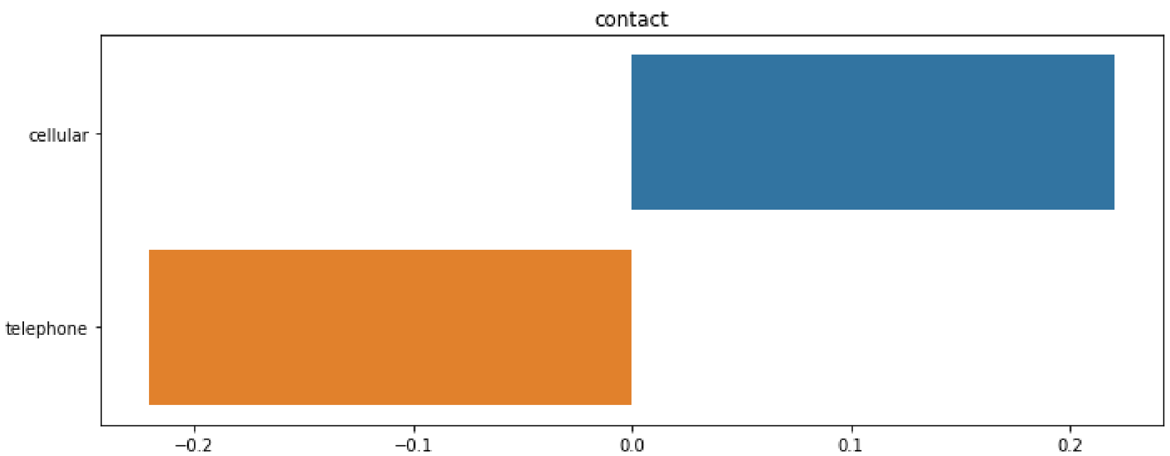


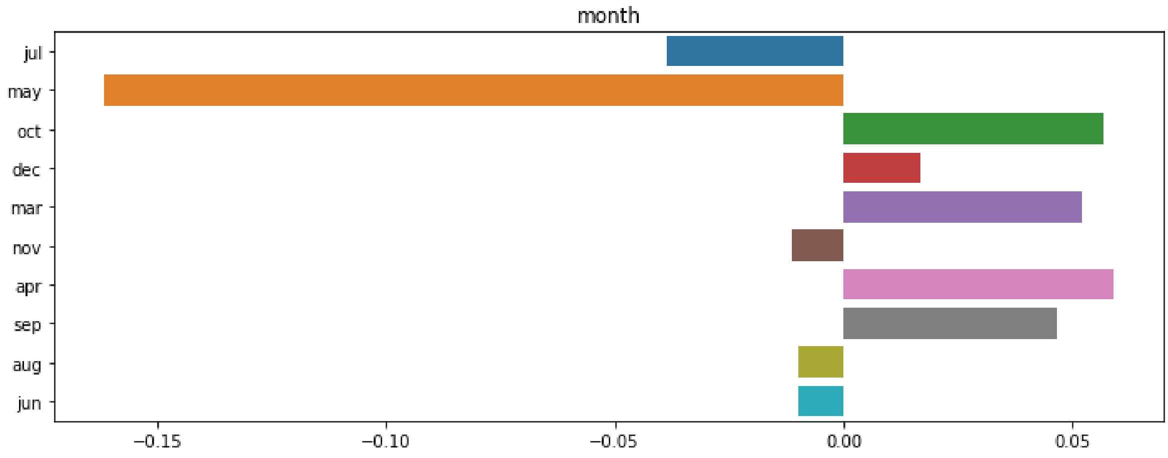

education
basic.4y
professional.course
basic.6y
illiterate
unknown
high.school
basic.9y
university.degree
−0.04
−0.02
0.00
0.02
0.04
0.06

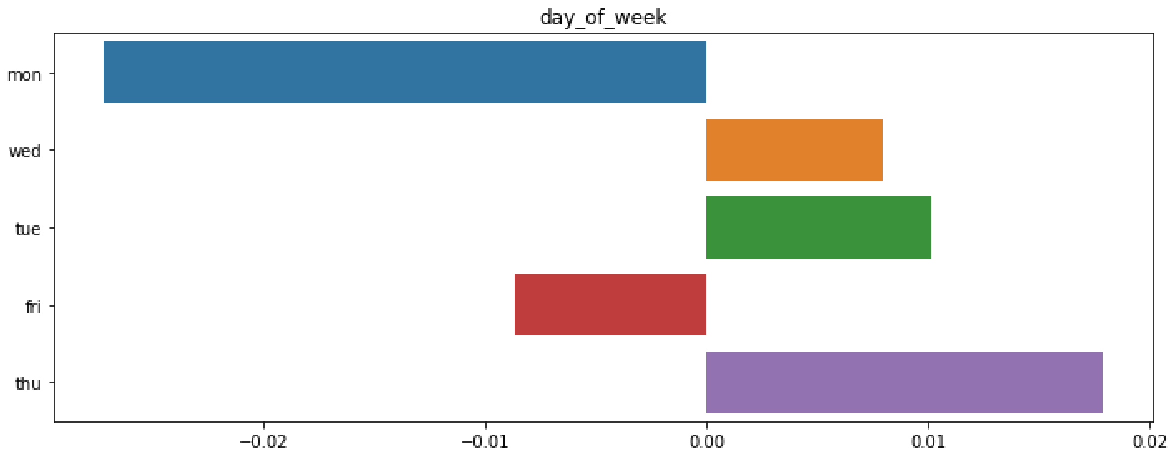

default
no
unknown
yes
−0.10
−0.05
0.00
0.05
0.10

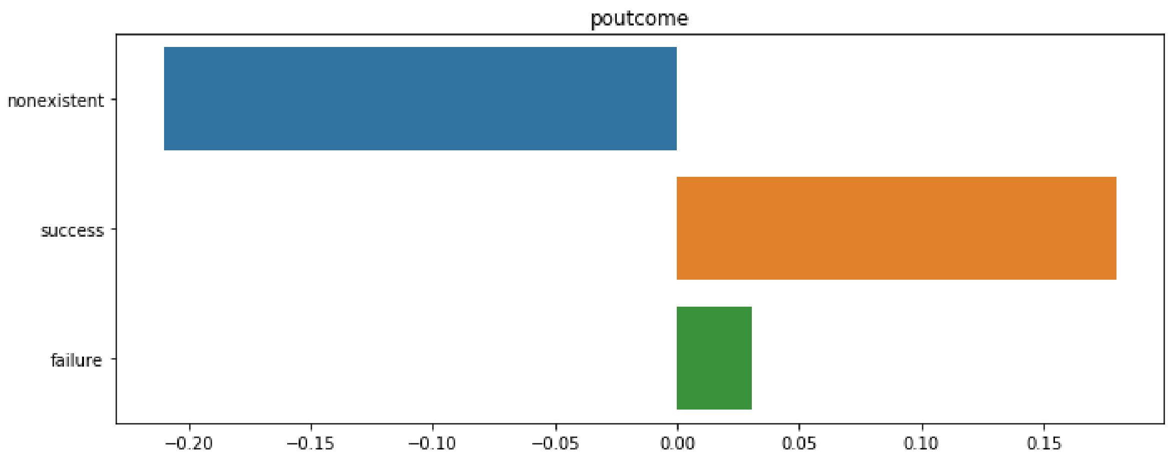

housing
no
unknown
yes
−0.015
−0.010
−0.005
0.000
0.005
0.010
0.015
0.020

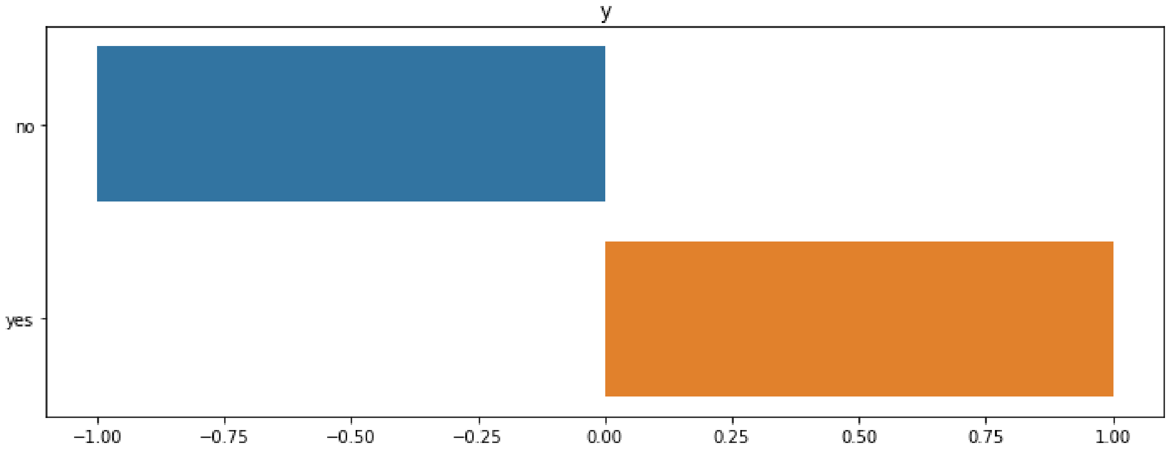

loan
no
unknown
yes
−0.004
−0.002
0.000
0.002
0.004
0.006


contact
cellular
telephone
−0.2
−0.1
0.0
0.1
0.2


month
jul
may
oct
dec
mar
nov
apr
sep
aug
jun
−0.15
−0.10
−0.05
0.00
0.05


day_of_week
mon
wed
tue
fri
thu
−0.02
−0.01
0.00
0.01
0.02

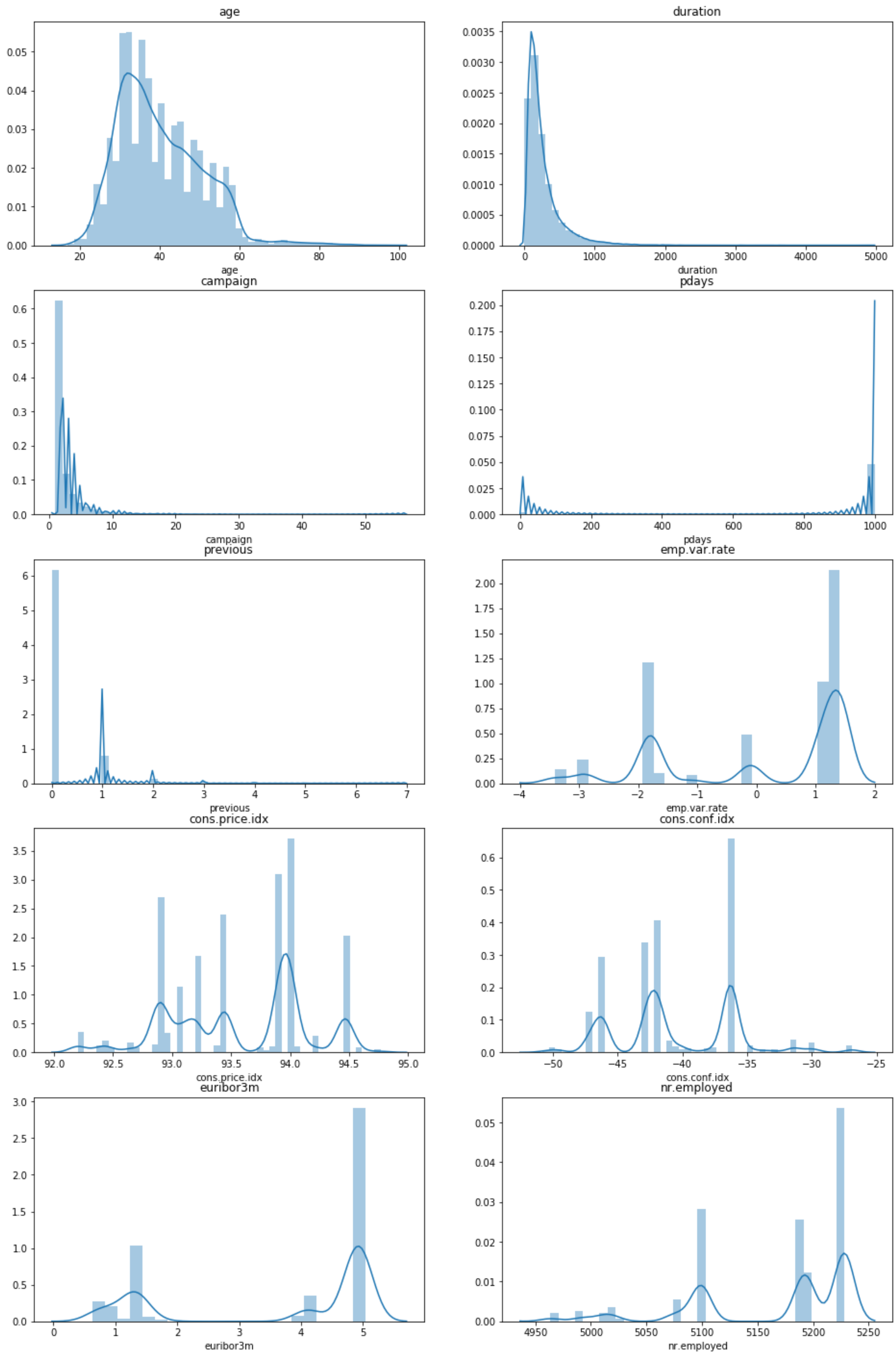

poutcome
nonexistent
success
failure
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


y
no
yes
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

In [14]:

```
k=1
fig = plt.figure(figsize = (16,30))
fig.suptitle("Various Bar Graph of Numerical Variable",fontsize=30)
for i in num:
    plt.subplot(6,2,k)
    sns.distplot(df[i])
    plt.title(str(i))

    k +=1
plt.show()
```

Various Bar Graph of Numerical Variable



In [15]:

```
# conisdering two variable and try to find out the insights
pd.crosstab(df.job,df.education)
```

Out[15]:

education	basic.4y	basic.6y	basic.9y	high.school	illiterate	professional.course	university.
job							
admin.	77	151	499	3329	1	363	
blue-collar	2318	1426	3623	878	8	453	
entrepreneur	137	71	210	234	2	135	
housemaid	474	77	94	174	1	59	
management	100	85	166	298	0	89	
retired	597	75	145	276	3	241	
self-employed	93	25	220	118	3	168	
services	132	226	388	2682	0	218	
student	26	13	99	357	0	43	
technician	58	87	384	873	0	3320	
unemployed	112	34	186	259	0	142	
unknown	52	22	31	37	0	12	

In [16]:

```
# Correlation Ananlysis
corr = df.corr()
```

In [17]:

```
corr
```

Out[17]:

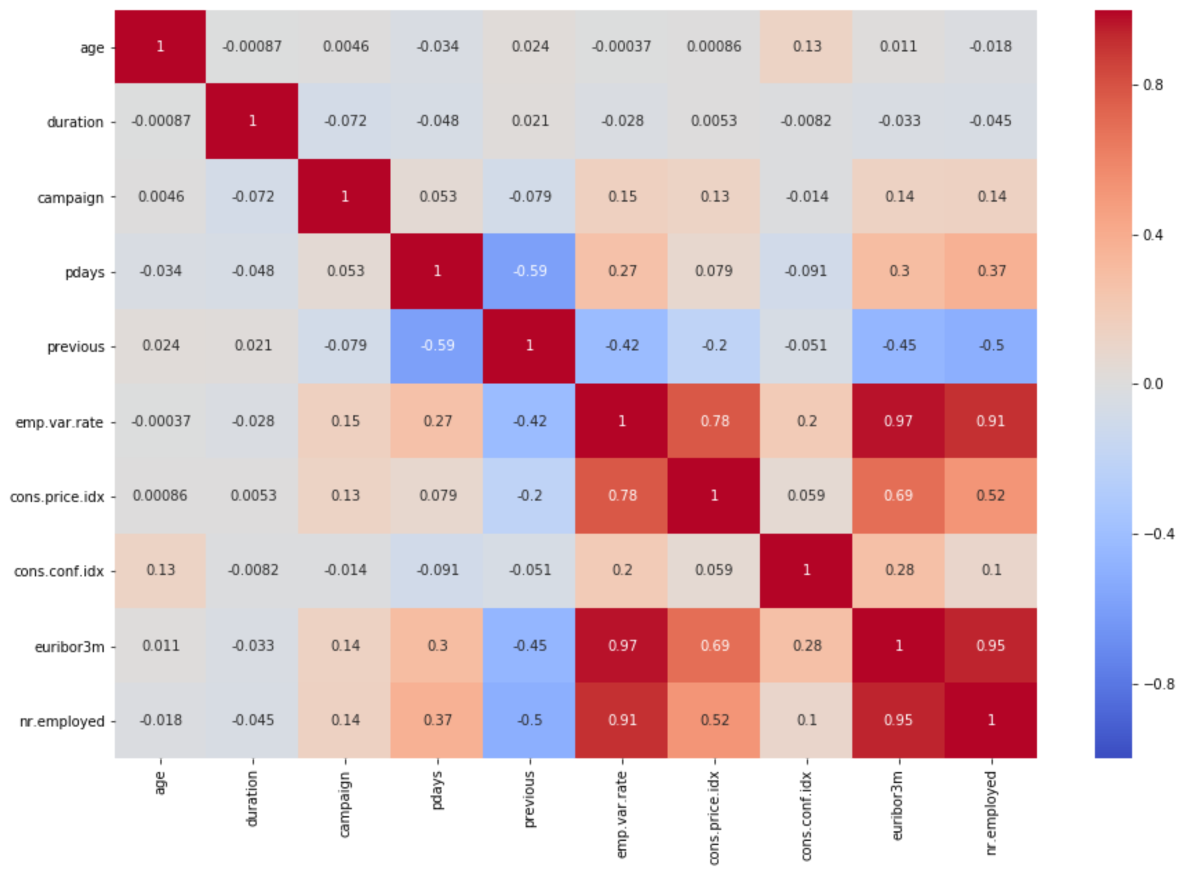
	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx
age	1.000000	-0.000866	0.004594	-0.034369	0.024365	-0.000371	0.000857
duration	-0.000866	1.000000	-0.071699	-0.047577	0.020640	-0.027968	0.005312
campaign	0.004594	-0.071699	1.000000	0.052584	-0.079141	0.150754	0.127836
pdays	-0.034369	-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 [18]:

```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),vmin=-1,cmap='coolwarm',annot=True)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2d21b045748>
```



In [19]:

```
df_dup = df[df.duplicated(keep="last")]
df_dup
```

Out[19]:

	age	job	marital	education	default	housing	loan	contact	month
1265	39	blue-collar	married	basic.6y	no	no	no	telephone	may
12260	36	retired	married	unknown	no	no	no	telephone	jul
14155	27	technician	single	professional.course	no	no	no	cellular	jul
16819	47	technician	divorced	high.school	no	yes	no	cellular	jul
18464	32	technician	single	professional.course	no	yes	no	cellular	jul
20072	55	services	married	high.school	unknown	no	no	cellular	aug
20531	41	technician	married	professional.course	no	yes	no	cellular	aug
25183	39	admin.	married	university.degree	no	no	no	cellular	nov
28476	24	services	single	high.school	no	yes	no	cellular	apr
32505	35	admin.	married	university.degree	no	yes	no	cellular	may
36950	45	admin.	married	university.degree	no	no	no	cellular	jul
38255	71	retired	single	university.degree	no	no	no	telephone	oct

12 rows × 21 columns

In [20]:

```
df_dup.shape
```

Out[20]:

```
(12, 21)
```

In [21]:

```
df = df.drop_duplicates()
df.shape
```

Out[21]:

```
(41176, 21)
```

In [22]:

```
# replacing differnet types of basic education with "basic"
df.replace(['basic.6y','basic.4y', 'basic.9y'], 'basic', inplace=True)
```

In [23]:

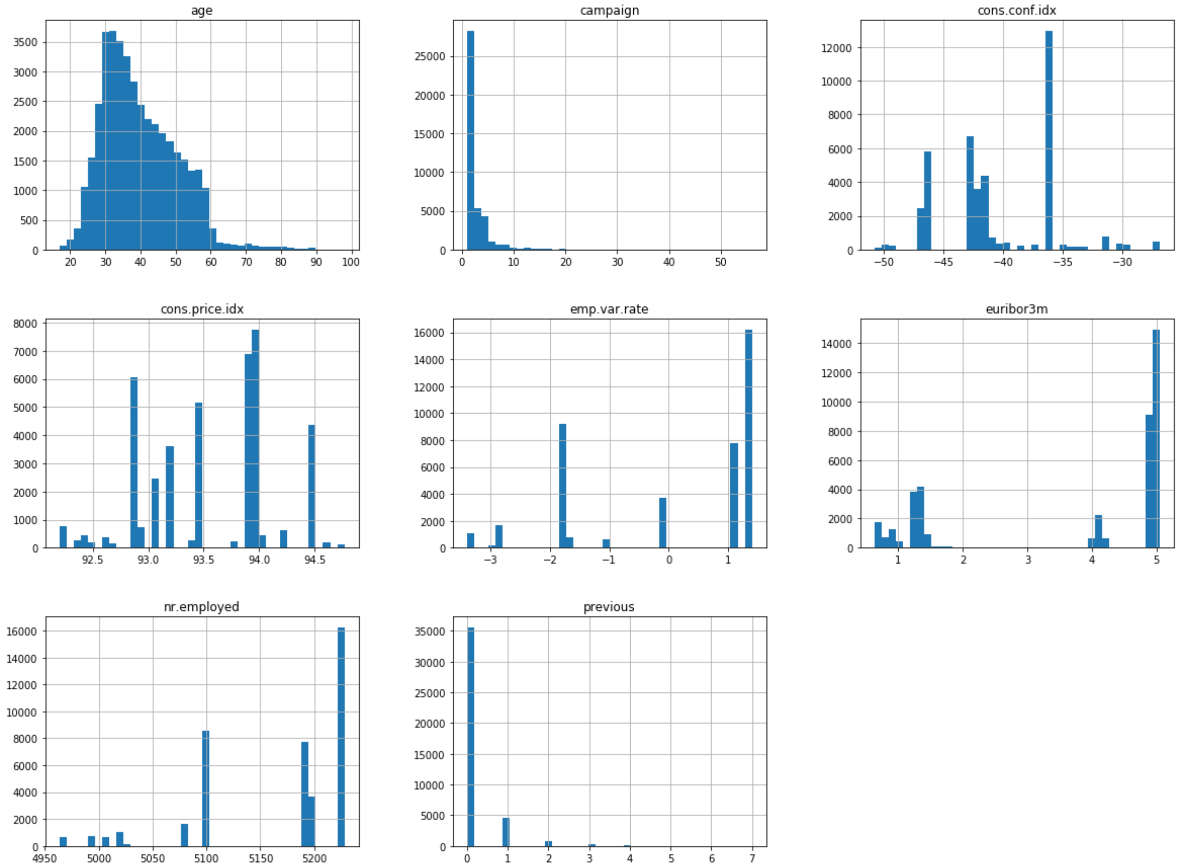
```
# dropping some of the unimportant variable to increase the accuracy
df.drop(['duration','contact','month','day_of_week','default','pdays',],axis=1,inplace=True
```

In [24]:

```
df.hist(bins=40,figsize=(20,15))
plt.show
```

Out[24]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```

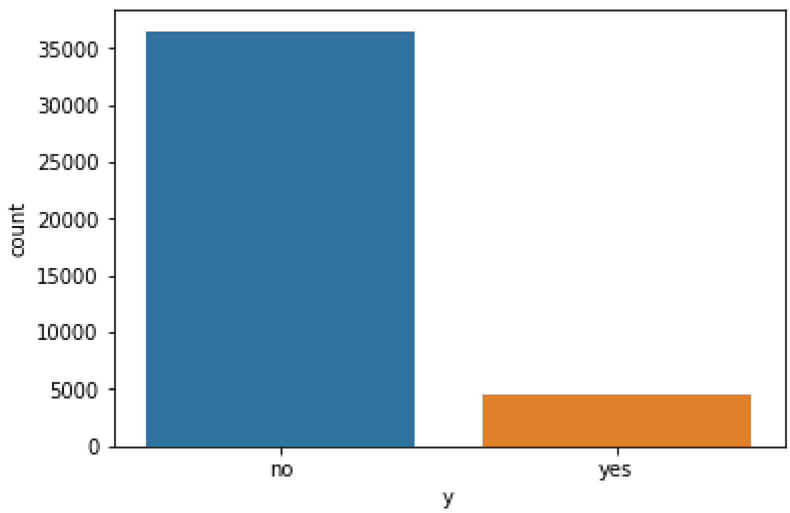


In [25]:

```
sns.countplot(df['y'])
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2d21a2c54e0>
```

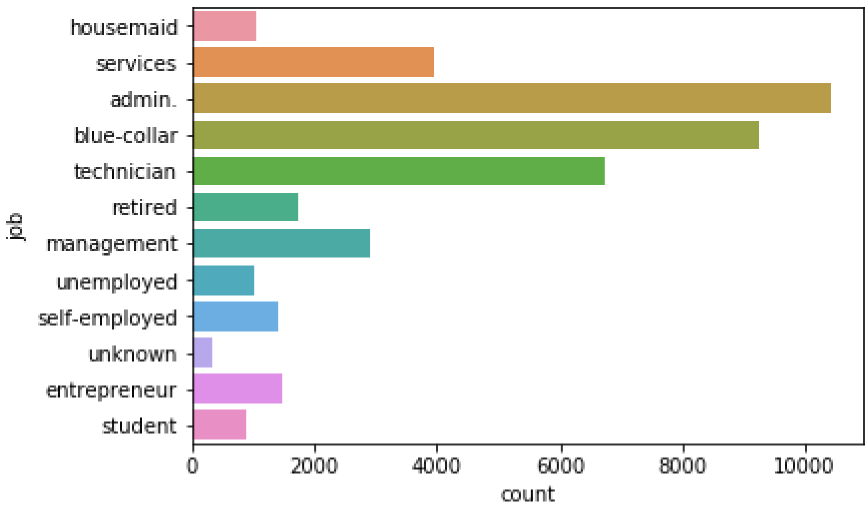


In [26]:

```
sns.countplot(y='job',data=df)
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2d21a416e80>
```

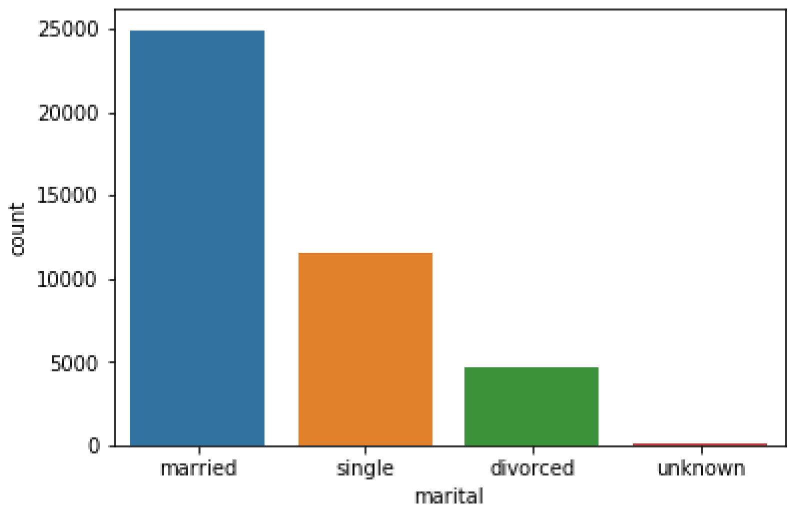


In [27]:

```
sns.countplot(df['marital'])
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2d21a49ae80>
```



In [28]:

```
# Converting Categorical variable into numeric using Label Encoder
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

In [29]:

```
df.job = le.fit_transform(df.job)
df.marital = le.fit_transform(df.marital)
df.education = le.fit_transform(df.education)
df.housing = le.fit_transform(df.housing)
df.loan = le.fit_transform(df.loan)
df.poutcome = le.fit_transform(df.poutcome)
```

In [30]:

```
df.y = le.fit_transform(df.y)
```

In [31]:

```
df.head()
```

Out[31]:

	age	job	marital	education	housing	loan	campaign	previous	poutcome	emp.var.rate	c
0	56	3	1	0	0	0	1	0	1	1.1	
1	57	7	1	1	0	0	1	0	1	1.1	
2	37	7	1	1	2	0	1	0	1	1.1	
3	40	0	1	0	0	0	1	0	1	1.1	
4	56	7	1	1	0	2	1	0	1	1.1	

In [32]:

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

In [33]:

```
# Train and Test split
from sklearn.model_selection import train_test_split
```

In [34]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [35]:

```
x_train.shape, y_train.shape
```

Out[35]:

```
((32940, 14), (32940,))
```

In [36]:

```
x_test.shape, y_test.shape
```

Out[36]:

```
((8236, 14), (8236,))
```

In [37]:

```
# Building Predictive Model
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [38]:

```
LR=LogisticRegression(max_iter=1000)
Dtree=DecisionTreeClassifier()
rfc=RandomForestClassifier(n_estimators=10)
clf = SVC(kernel='rbf', gamma='auto')

def accuracy (a,b,c,d):
    for every in (a,b,c,d):
        every.fit(x_train,y_train)
        print(every.__class__.__name__,'accuracy_score=',accuracy_score(y_test,every.predic
accuracy(LR,Dtree,rfc,clf)
```

```
C:\Users\bansa\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)

LogisticRegression accuracy_score= 0.8949732880038854
DecisionTreeClassifier accuracy_score= 0.8442204953861098
RandomForestClassifier accuracy_score= 0.8863525983487129
SVC accuracy_score= 0.8870811073336571
```

In [39]:

```
from sklearn.metrics import classification_report
yhat = LR.predict(x_test)
print(classification_report(y_test,yhat))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      7269
           1       0.76      0.15      0.26       967

   micro avg       0.89      0.89      0.89      8236
   macro avg       0.83      0.57      0.60      8236
weighted avg       0.88      0.89      0.86      8236
```

In [40]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, yhat)
print(confusion_matrix)
```

```
[[7222   47]
 [ 818  149]]
```

In [ ]: