Data Visualisations

This notebook is in continuation with the previous `first-insights.ipynb` notebook

This notebook aims at further exploration of data using data visualisation techniques. The numerical exploration done in the previous notebook gave us some useful insigts about the data and we were able to draw large number of conclusion about features and the trend they tend to follow.

However, most of the times it is better to use visualisations to analyse data across multiple dimensions

The first cell contains dependencies needed in this notebook

In [6]:

```
import numpy as np #Matrix-Maths
import pandas as pd #DataFrame
import seaborn as sns #Advanced Visualisation
from matplotlib import pyplot as plt #matlab style plotting

%matplotlib inline
```

In [8]:

```
data = pd.read_csv('Data/bank-additional-full.csv', delimiter=';')
```

In [9]:

```
data.describe()
```

Out[9]:

	age	duration	campaign	pdays	previous	emp.var.rate
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
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

As stated in the previous notebook, we need to convert y reponses to binary numerical figures

In [10]:

```
data['y'] = data['y'].replace('yes', 1)
data['y'] = data['y'].replace('no', 0)
```

In [11]:

```
for i in ['job', 'marital', 'education', 'default', 'housing', 'loan']:
    data.drop(data.loc[data[i]=='unknown'].index, inplace=True)

data = data.reset_index(drop=True)
```

In [12]:

```
data.head()
```

Out[12]:

	age	job	marital	education	default	housing	loan	contact	month	day
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	37	services	married	high.school	no	yes	no	telephone	may	
2	40	admin.	married	basic.6y	no	no	no	telephone	may	
3	56	services	married	high.school	no	no	yes	telephone	may	
4	59	admin.	married	professional.course	no	no	no	telephone	may	

5 rows × 21 columns

In [13]:

```
count_yes = len(data[data['y']==1])
count_no = len(data[data['y']==0])
print('Count of Yes: ', count_yes)
print('Count of No: ', count_no)
print('Sum of both the counts: ', count_yes+count_no)
print('Total number of datapoints: ', len(data['y']))
```

```
Count of Yes:  3859
Count of No:  26629
Sum of both the counts:  30488
Total number of datapoints:  30488
```

In [14]:

```
data = data.drop('duration', axis=1)
```

Some basic preprocessing is applied above taking ideas from the previous notebook

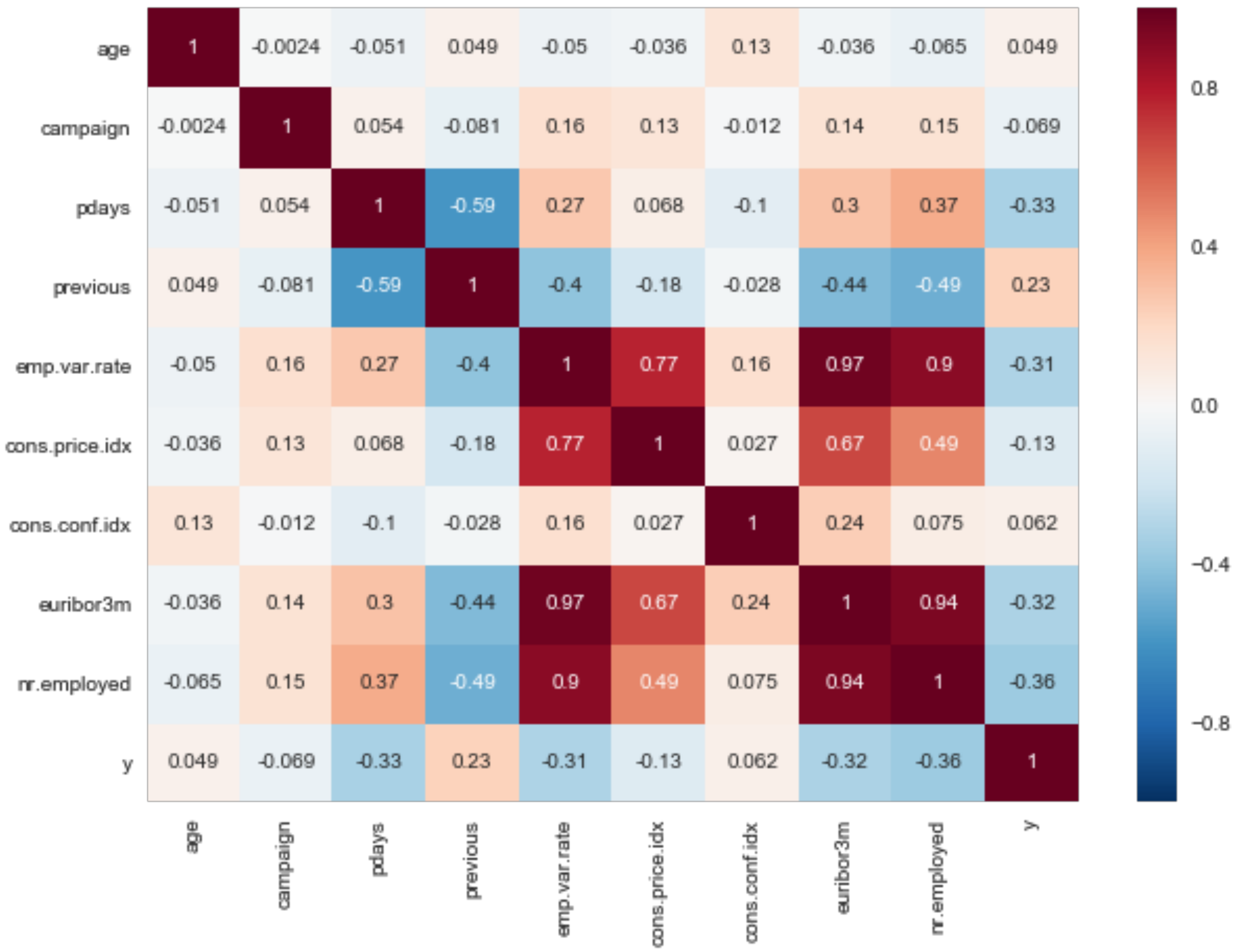
The Visualisations and corresponding conclusions is displayed below

In [15]:

```
data_corr = data.corr(method='pearson')
plt.figure(figsize=(10, 7))
sns.heatmap(data_corr, annot=True)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x113eb8978>
```



The output variable y shows maximum positive correlation with previous feature and most negative correlation with nr.employed. euribor3m also seem to have high negative correlation with euribor3m. This correlation heatmap is drawn by only taking into account the numeric features. A more detailed heatmap including categorical features as well below after preliminary analysis

The above heatmap also shows that emp.var.rate and euboir3m is highly related and one of them can be ignored. Also, nr.employed is highly correlated to euboir3m. Thus, this feature can be safely removed without loss of information. Similar is the case with `nr.employed` and `cons.price.idx`

In [16]:

```
data = data.drop('nr.employed', axis=1)
data = data.drop('cons.price.idx', axis=1)
```

In [17]:

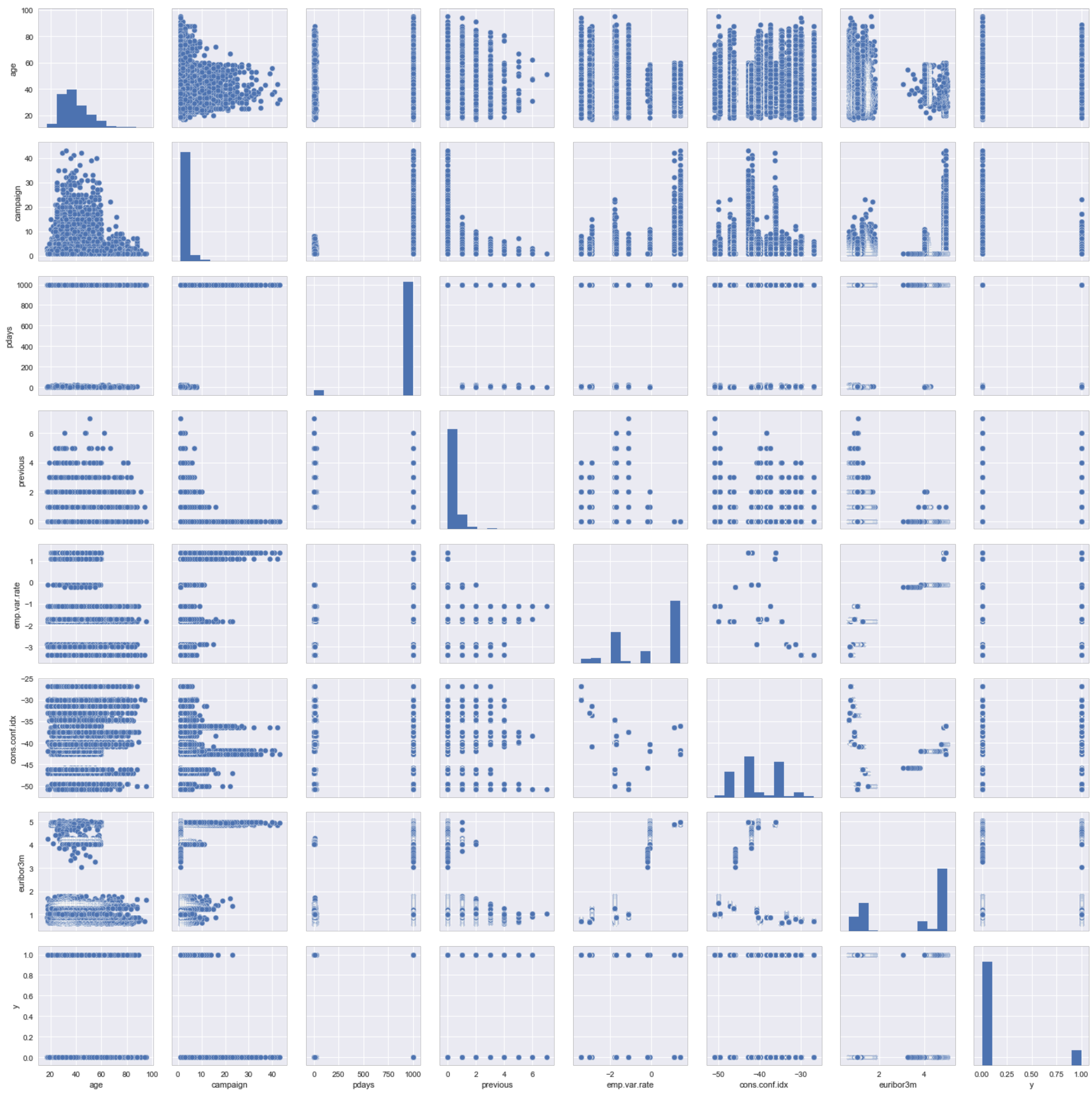
```
cols = list(data.columns)
```

In [18]:

```
sns.pairplot(data)
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x11487c198>
```



The pairplot shown above gives scatter plots of different numerical features with histograms in diagonal. This gives a rough basic variation of different parameters with each other.

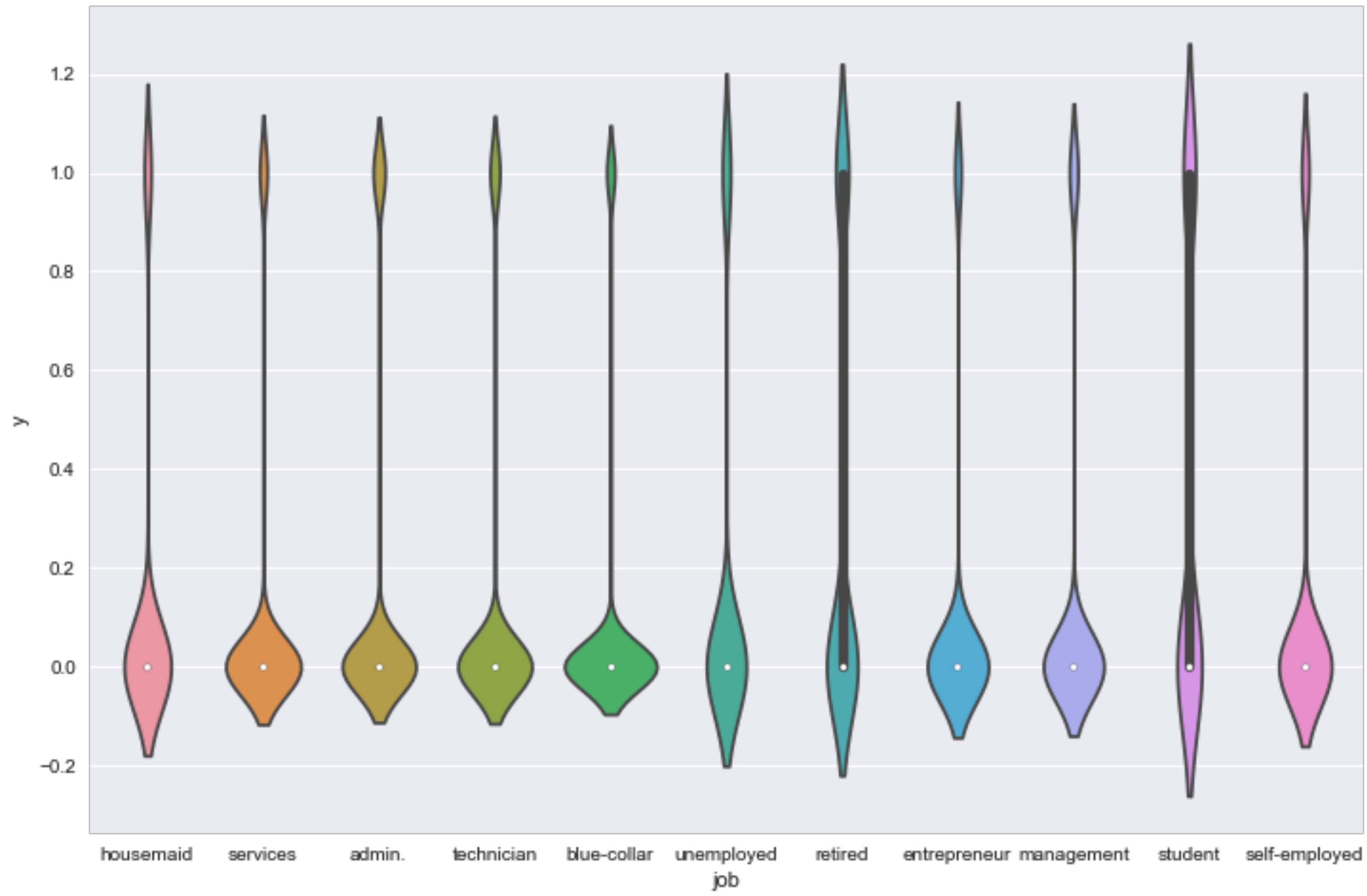
The violin plots shown below shows distribution of categorical data in different classes. It is clearly observed by the width of the plot that most of the reponses are negative and very few responses are positive in each category.

In [19]:

```
plt.figure(figsize=(12, 8))
sns.violinplot(x='job', y='y', data=data)
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x11a3f26d8>

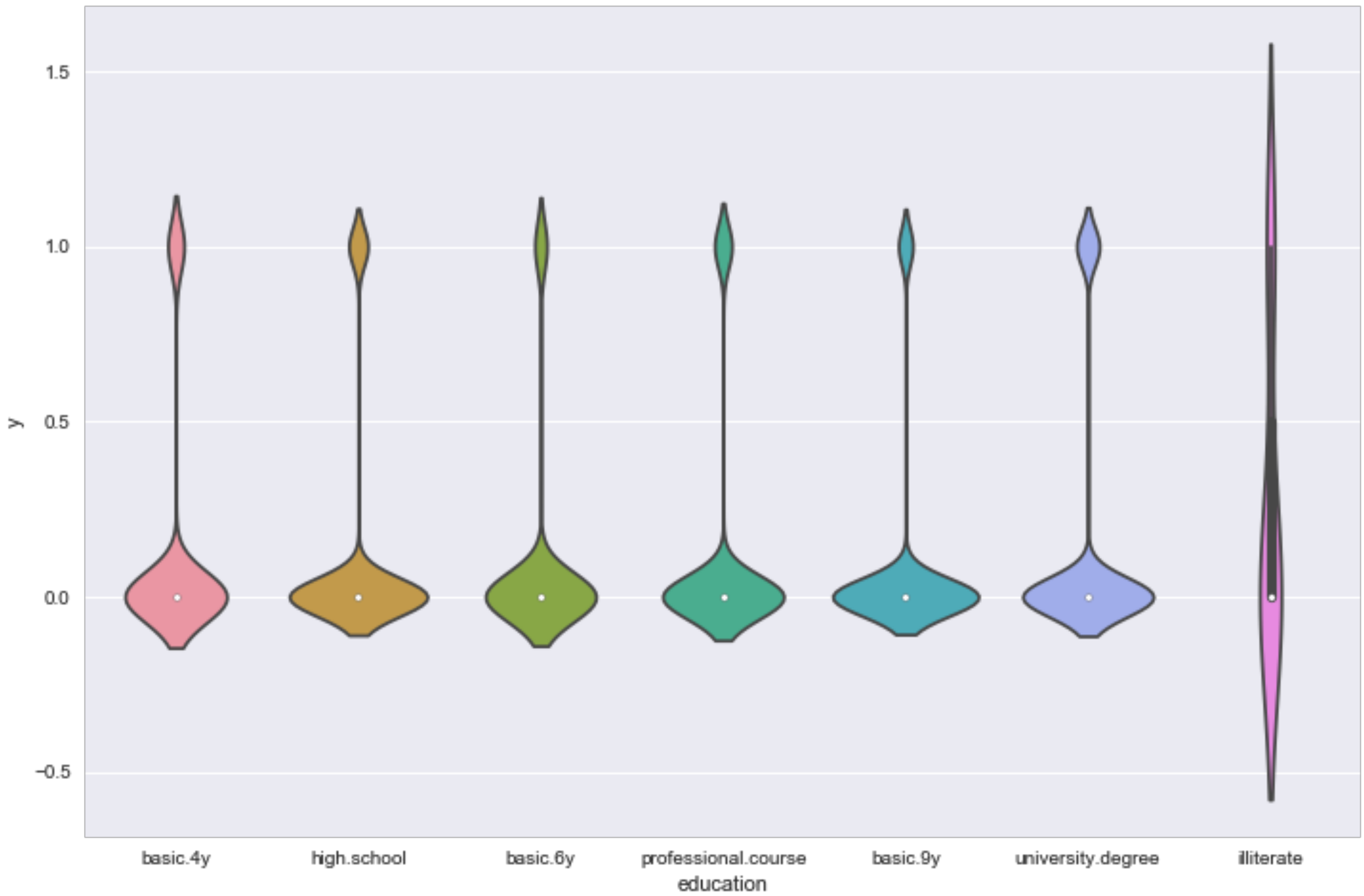


In [20]:

```
plt.figure(figsize=(12, 8))
sns.violinplot(x='education', y='y', data=data)
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x115530e80>
```



The function below is a helper function which is used in ploting based on positive and negative responses. The plots below displays how the reponse is distributed based on various categorival values

In [26]:

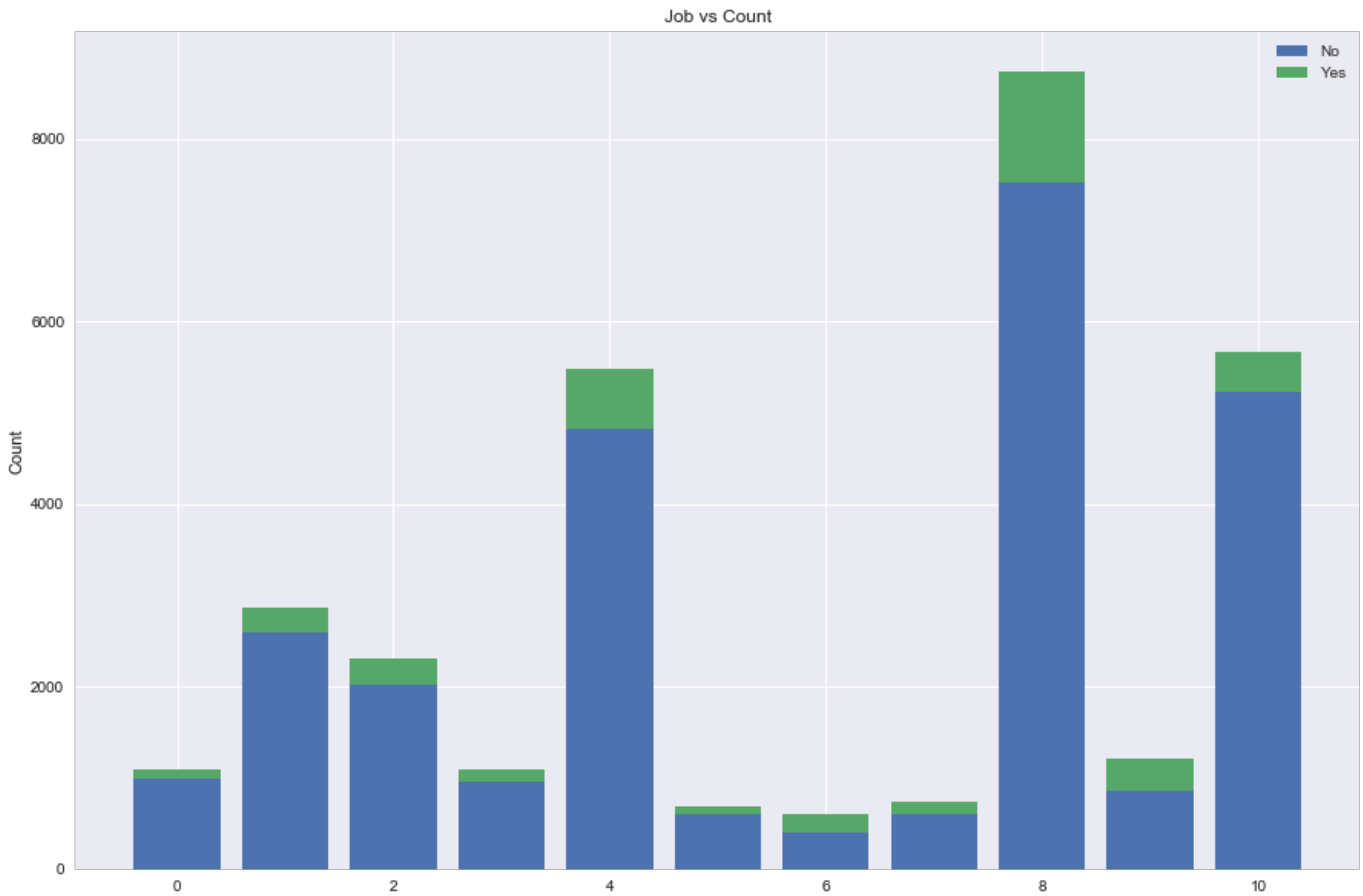
```
def make_categorical_barplot_vs_y(x, df, y = 'y'):
    local_type = list(set(df[x].values))
    df_jy = df[[x, y]]
    clus_ = {i: {'no': 0, 'yes': 1} for i in local_type}
    for i in range(len(df_jy[x])):
        yes = df_jy[y].values[i]
        loc = df_jy[x].values[i]
        if yes == 1:
            clus_[loc]['yes'] += 1
        else:
            clus_[loc]['no'] += 1

    y_ = [clus_[c]['yes'] for c in clus_]
    n_ = [clus_[c]['no'] for c in clus_]

    return list(clus_), y_, n_
```

In [36]:

```
x, y_, n_ = make_categorical_barplot_vs_y(x = 'job',df = data)
x = np.arange(len(x))
plt.figure(figsize = (15, 10))
p1 = plt.bar(x, n_)
p2 = plt.bar(x, y_, bottom = n_)
plt.legend((p1[0], p2[0]), ('No', 'Yes'))
plt.ylabel('Count')
plt.title('{0} vs Count'.format('Job'))
```

Out[36]:

```
<matplotlib.text.Text at 0x11d7c33c8>
```



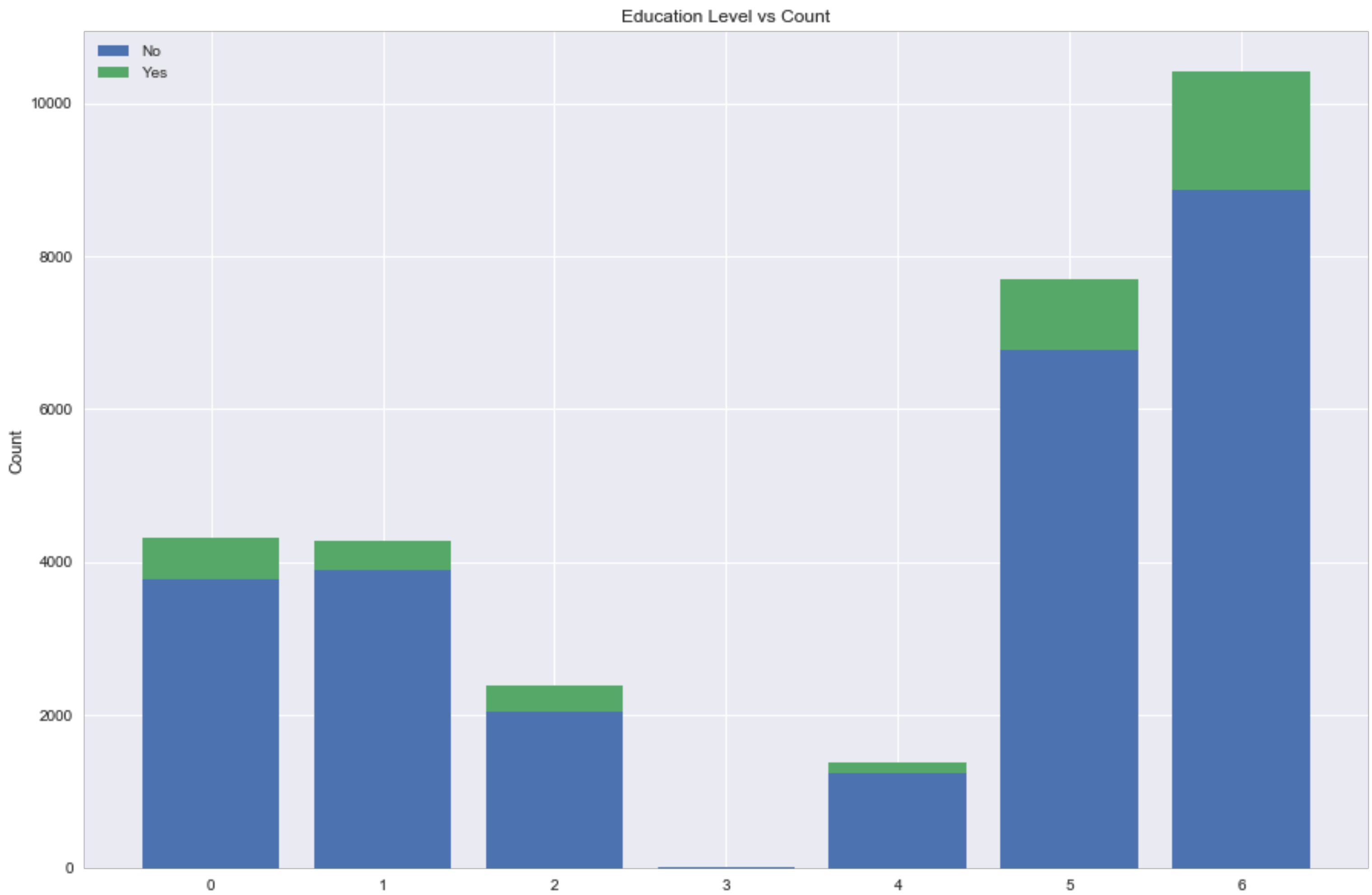
LEGEND:
0: entrepreneur
1: services
2: management
3: self-employed
4: technician
5: housemaid
6: student
7: unemployed
8: admin.
9: retired
10: blue-collar

In [41]:

```
x, y_, n_ = make_categorical_barplot_vs_y(x = 'education',df = data)
x = np.arange(len(x))
plt.figure(figsize = (15, 10))
p1 = plt.bar(x, n_)
p2 = plt.bar(x, y_, bottom = n_)
plt.legend((p1[0], p2[0]), ('No', 'Yes'))
plt.ylabel('Count')
plt.title('{0} vs Count'.format('Education Level'))
```

Out[41]:

```
<matplotlib.text.Text at 0x11dd6a160>
```




```
LEGEND:
0: professional.course
1: basic.9y
2: basic.4y
3: illiterate
4: basic.6y
5: high.school
6: university.degree
```

The visualisations above gave distinctive feature outcomes for both numerical and categorical values. This and first-insights togather completes our data exploration task. The next and final task is building a predictive model which is done in the next notebook

In [ ]:

```

```

Data Visualisations

This notebook is in continuation with the previous `first-insights.ipynb` notebook

This notebook aims at further exploration of data using data visualisation techniques. The numerical exploration done in the previous notebook gave us some useful insigts about the data and we were able to draw large number of conclusion about features and the trend they tend to follow.

However, most of the times it is better to use visualisations to analyse data across multiple dimensions

The first cell contains dependencies needed in this notebook

In [6]:

```

```

In [8]:

```

```

In [9]:

```

```

Out[9]:

	age	duration	campaign	pdays	previous	emp.var.rate
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
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

As stated in the previous notebook, we need to convert y reponses to binary numerical figures

In [10]:

In [11]:

In [12]:

Out[12]:

	age	job	marital	education	default	housing	loan	contact	month	day
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	37	services	married	high.school	no	yes	no	telephone	may	
2	40	admin.	married	basic.6y	no	no	no	telephone	may	
3	56	services	married	high.school	no	no	yes	telephone	may	
4	59	admin.	married	professional.course	no	no	no	telephone	may	

5 rows × 21 columns

In [13]:

```
Count of Yes:  3859
Count of No:  26629
Sum of both the counts:  30488
Total number of datapoints:  30488
```

In [14]:

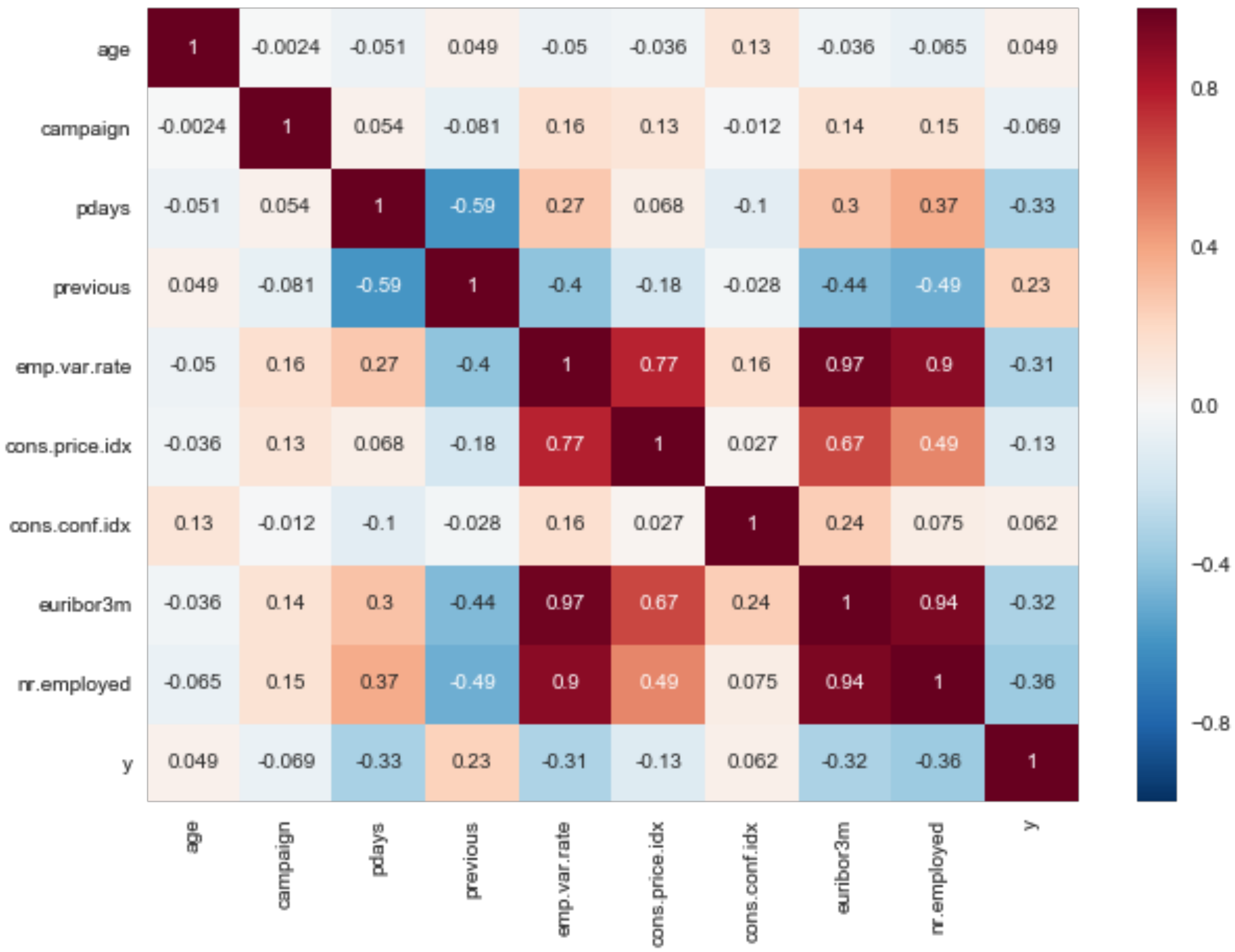
Some basic preprocessing is applied above taking ideas from the previous notebook

The Visualisations and corresponding conclusions is displayed below

In [15]:

Out[15]:

<matplotlib.axes._subplots.AxesSubplot at 0x113eb8978>



The output variable y shows maximum positive correlation with previous feature and most negative correlation with nr.employed. euribor3m also seem to have high negative correlation with euribor3m. This correlation heatmap is drawn by only taking into account the numeric features. A more detailed heatmap including categorical features as well below after preliminary analysis

The above heatmap also shows that emp.var.rate and euboir3m is highly related and one of them can be ignored. Also, nr.employed is highly correlated to euboir3m. Thus, this feature can be safely removed without loss of information. Similar is the case with `nr.employed` and `cons.price.idx`

In [16]:

In [17]:

In [18]:

```

```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x11487c198>
```

The pairplot shown above gives scatter plots of different numerical features with histograms in diagonal. This gives a rough basic variation of different parameters with each other.

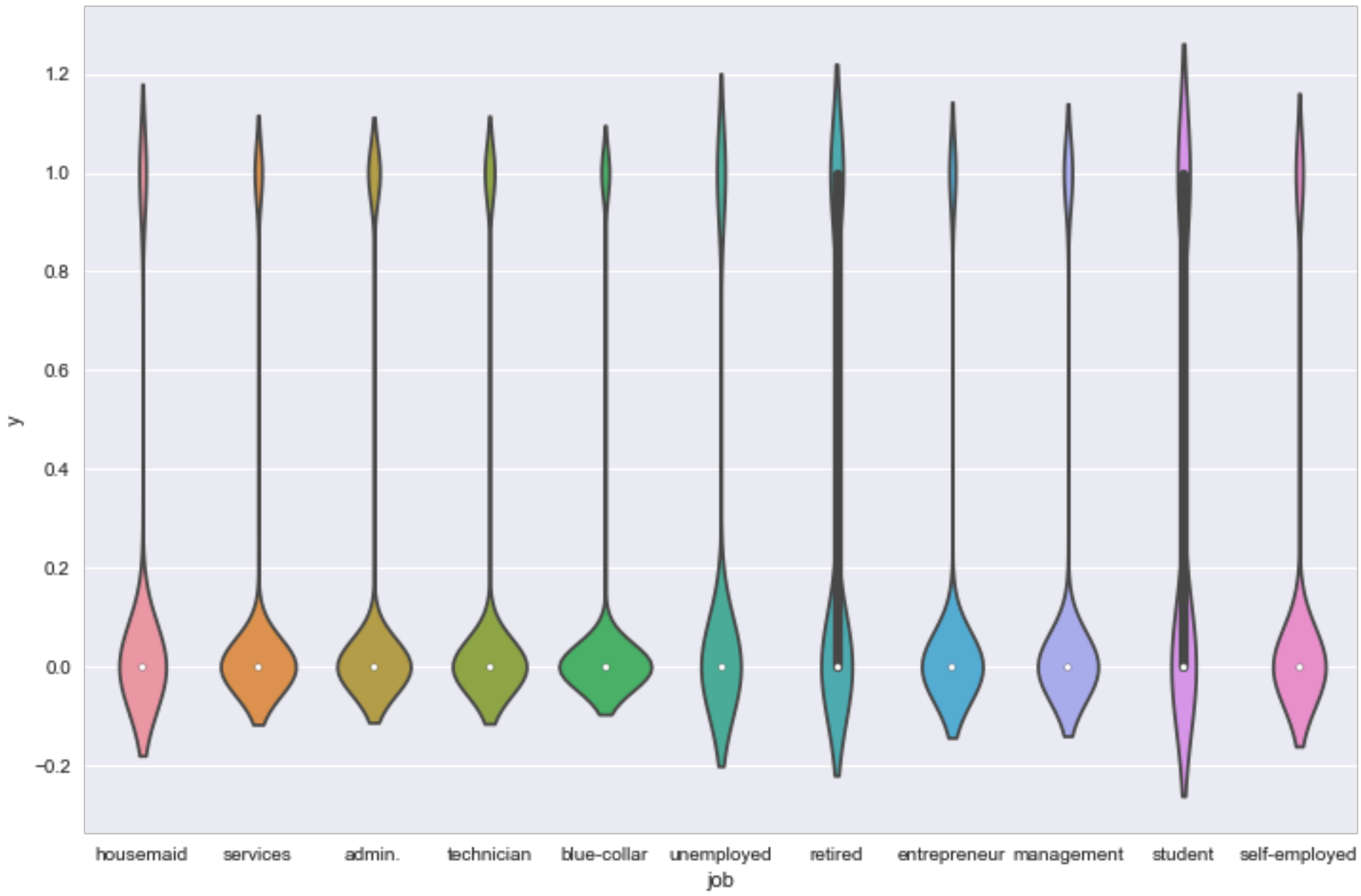
The violin plots shown below shows distribution of categorical data in different classes. It is clearly observed by the width of the plot that most of the reponses are negative and very few responses are positive in each category.

In [19]:

```

```

Out[19]:

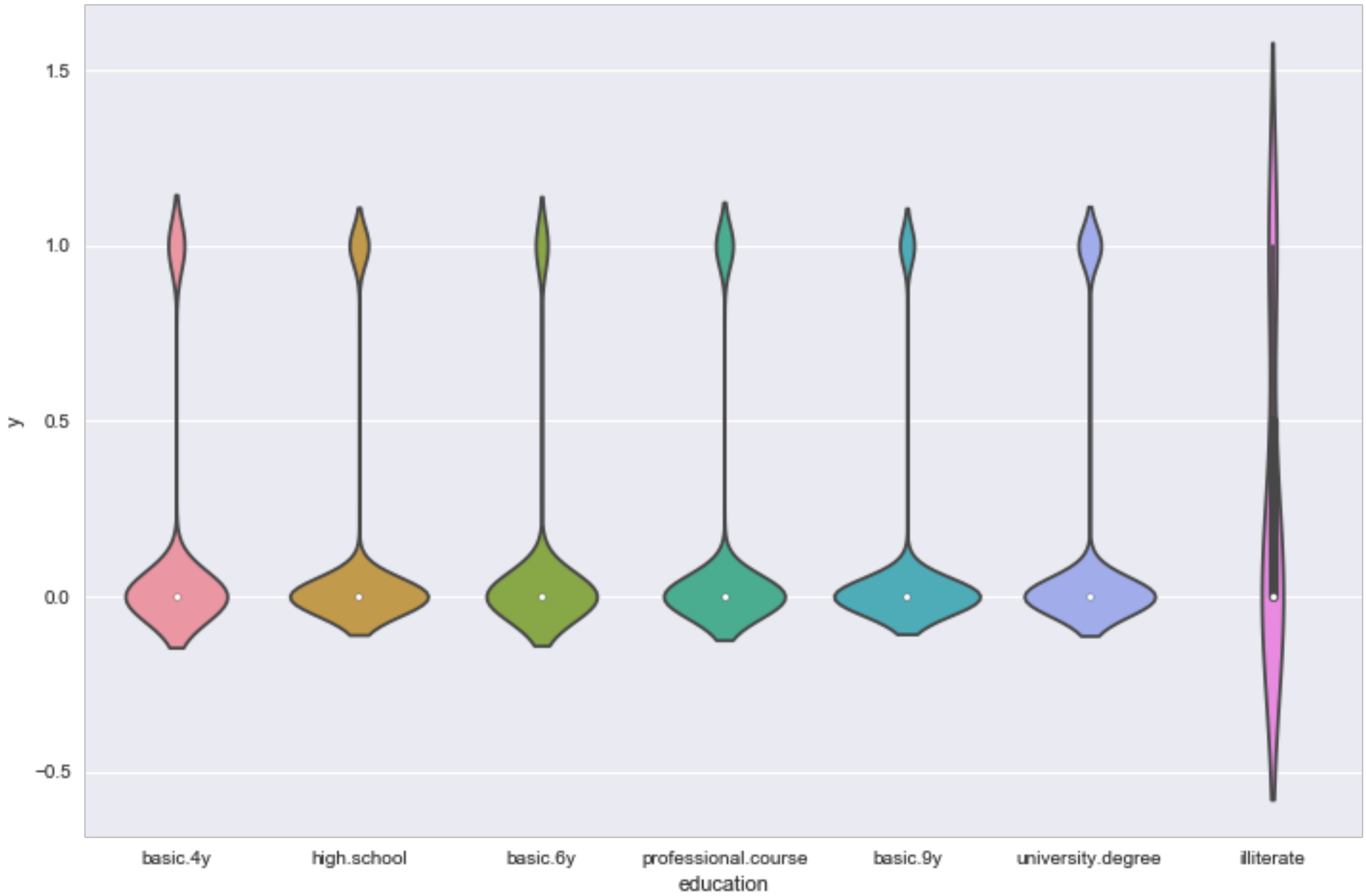
```
<matplotlib.axes._subplots.AxesSubplot at 0x11a3f26d8>
```



In [20]:

Out[20]:

<matplotlib.axes._subplots.AxesSubplot at 0x115530e80>



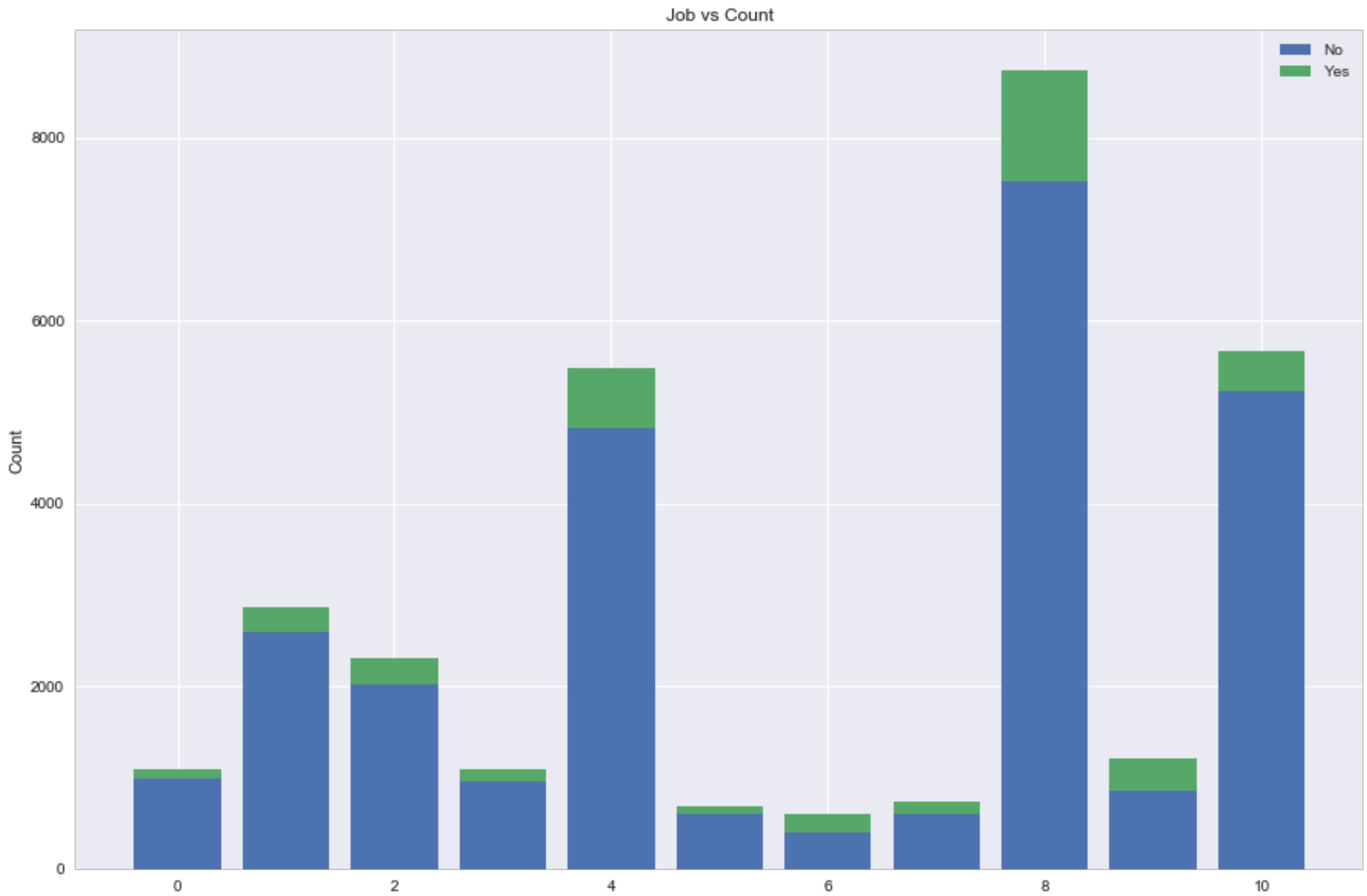
The function below is a helper function which is used in ploting based on positive and negative responses. The plots below displays how the reponse is distributed based on various categorival values

In [26]:

In [36]:

Out[36]:

```
<matplotlib.text.Text at 0x11d7c33c8>
```

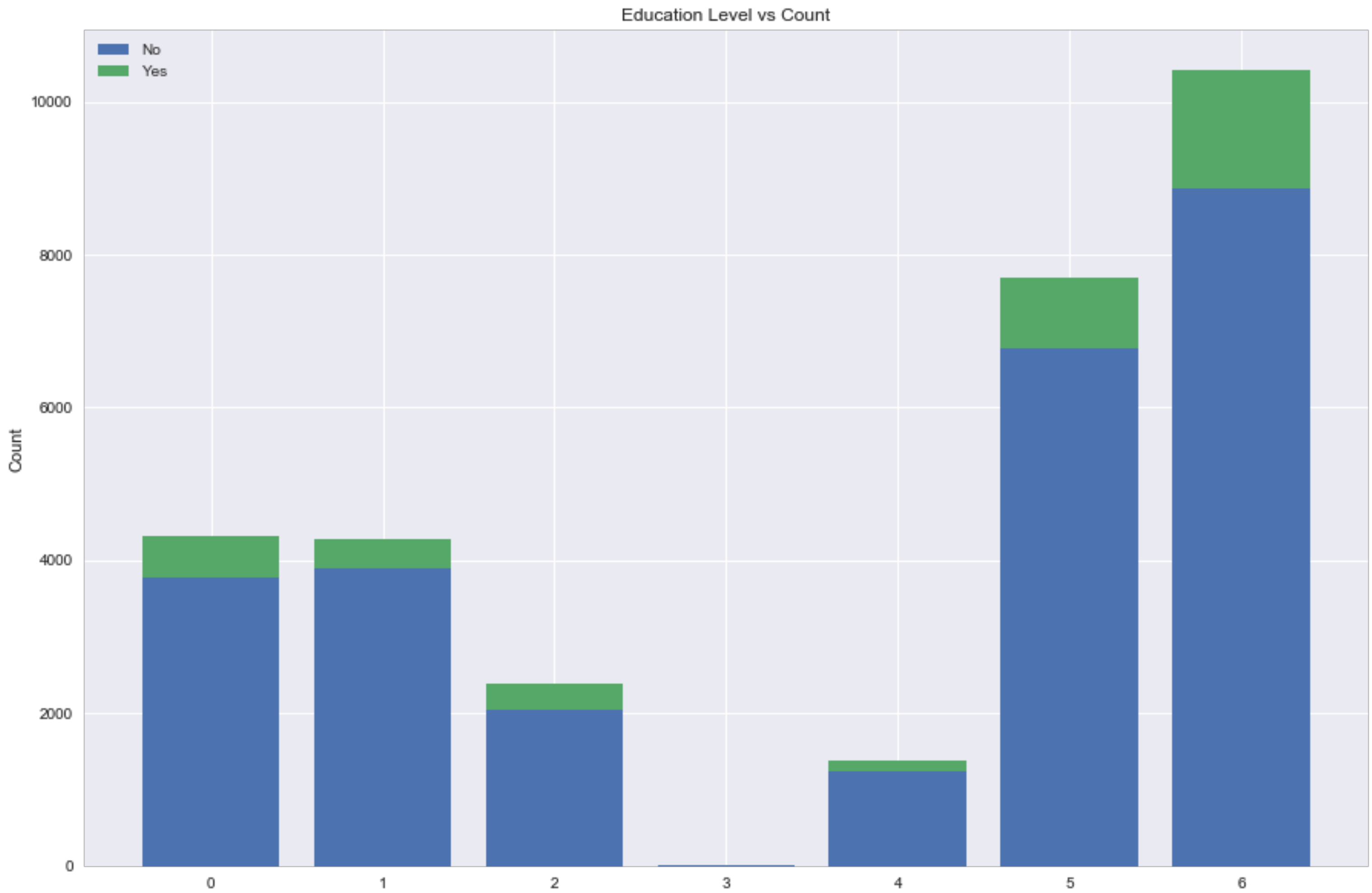



```
LEGEND:
0: entrepreneur
1: services
2: management
3: self-employed
4: technician
5: housemaid
6: student
7: unemployed
8: admin.
9: retired
10: blue-collar
```

In [41]:

Out[41]:

<matplotlib.text.Text at 0x11dd6a160>




```
LEGEND:
0: professional.course
1: basic.9y
2: basic.4y
3: illiterate
4: basic.6y
5: high.school
6: university.degree
```

The visualisations above gave distinctive feature outcomes for both numerical and categorical values. This and first-insights togather completes our data exploration task. The next and final t