In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [4]:
```python
df = pd.read_csv("bank-additional-full.csv",delimiter=';')
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.c
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	

5 rows × 21 columns

In [5]:
```python
df.head()
```

Out[5]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.c
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	

5 rows × 21 columns

In [6]:
```python
df.tail()
```

Out[6]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0	nonexistent	-1.1	94.767
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	-1.1	94.767
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0	nonexistent	-1.1	94.767
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	-1.1	94.767
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1	failure	-1.1	94.767

5 rows × 21 columns

In [7]:
```python
df.columns
```

Out[7]:
```
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', 'deposit'],
      dtype='object')
```

In [8]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  deposit         41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

In [9]:
```python
df.shape
```

Out[9]: (41188, 21)

In [10]:
```python
df.dtypes
```

Out[10]:
```
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
deposit           object
dtype: object
```

In [12]:
```python
df.dtypes.value_counts()
```

Out[12]:
```
object     11
int64       5
float64     5
Name: count, dtype: int64
```

In [13]:
```python
df.duplicated().sum()
```

Out[13]: np.int64(12)

In [14]:
```python
df.isna().sum()
```

Out[14]:
```
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
deposit           0
dtype: int64
```

In [15]:
```python
cat_cols = df.select_dtypes(include='object').columns
print(cat_cols)

num_cols = df.select_dtypes(exclude='object').columns
print(num_cols)
```

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

In [16]:
```python
df.describe()
```

Out[16]:

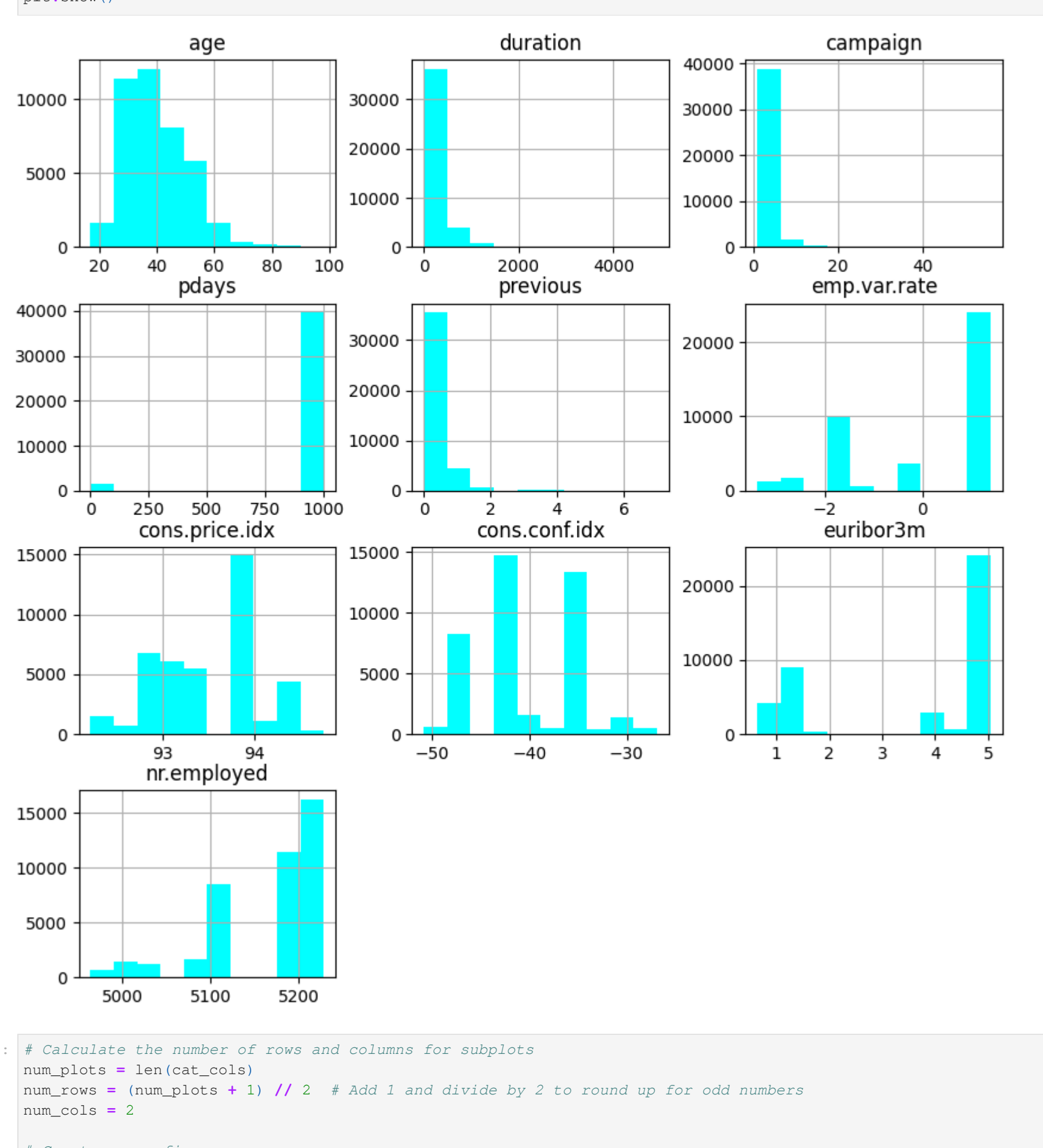
	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

In [17]:
```python
df.describe(include='object')
```

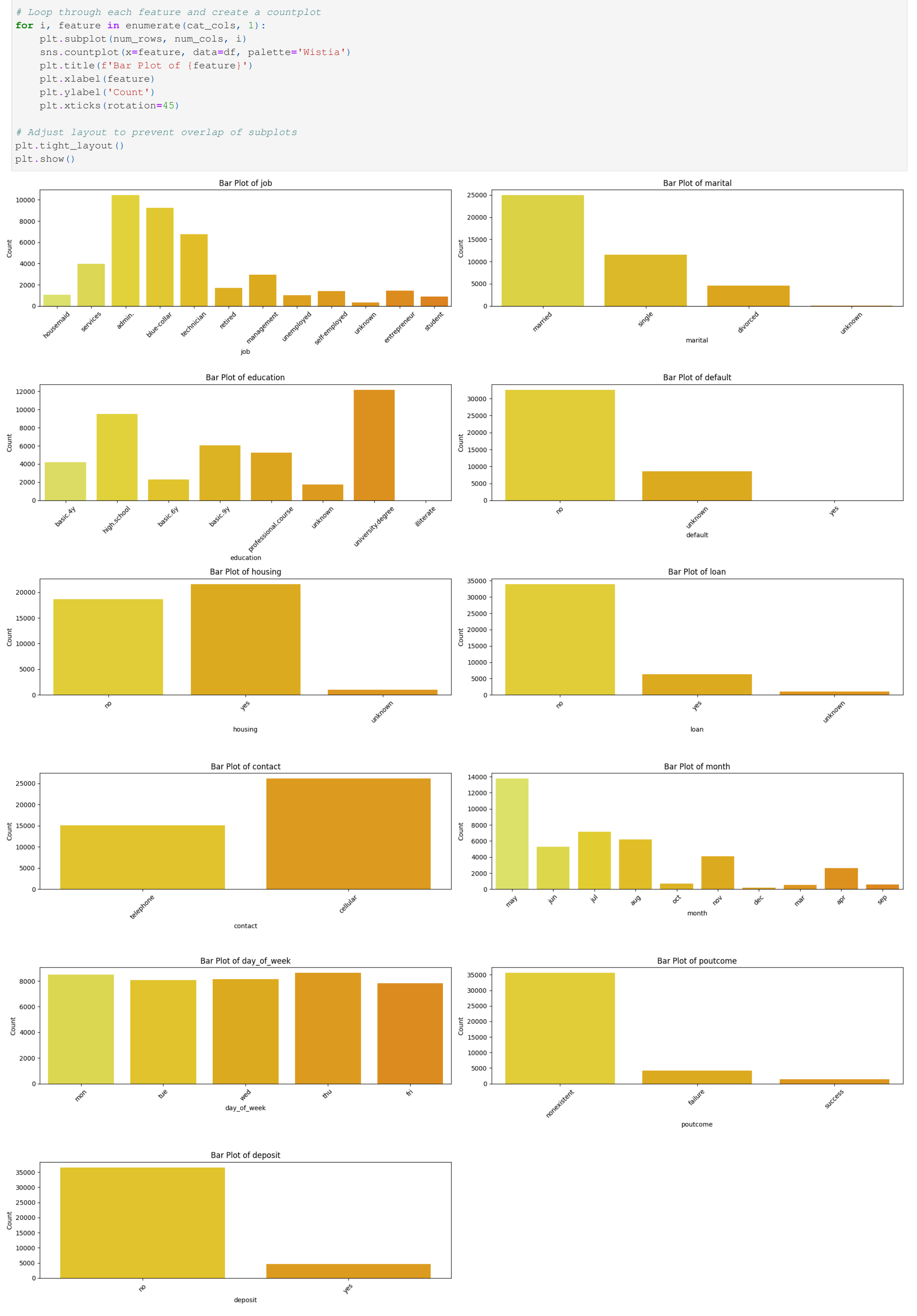
Out[17]:

	job	marital	education	default	housing	loan	contact	month	day_of_week	poutcome	deposit
count	41188	41188	41188	41188	41188	41188	41188	41188	41188	41188	41188
unique	12	4	8	3	3	3	2	10	5	3	2
top	admin.	married	university.degree	no	yes	no	cellular	may	thu	nonexistent	no
freq	10422	24928	12168	32588	21576	33950	26144	13769	8623	35563	36548

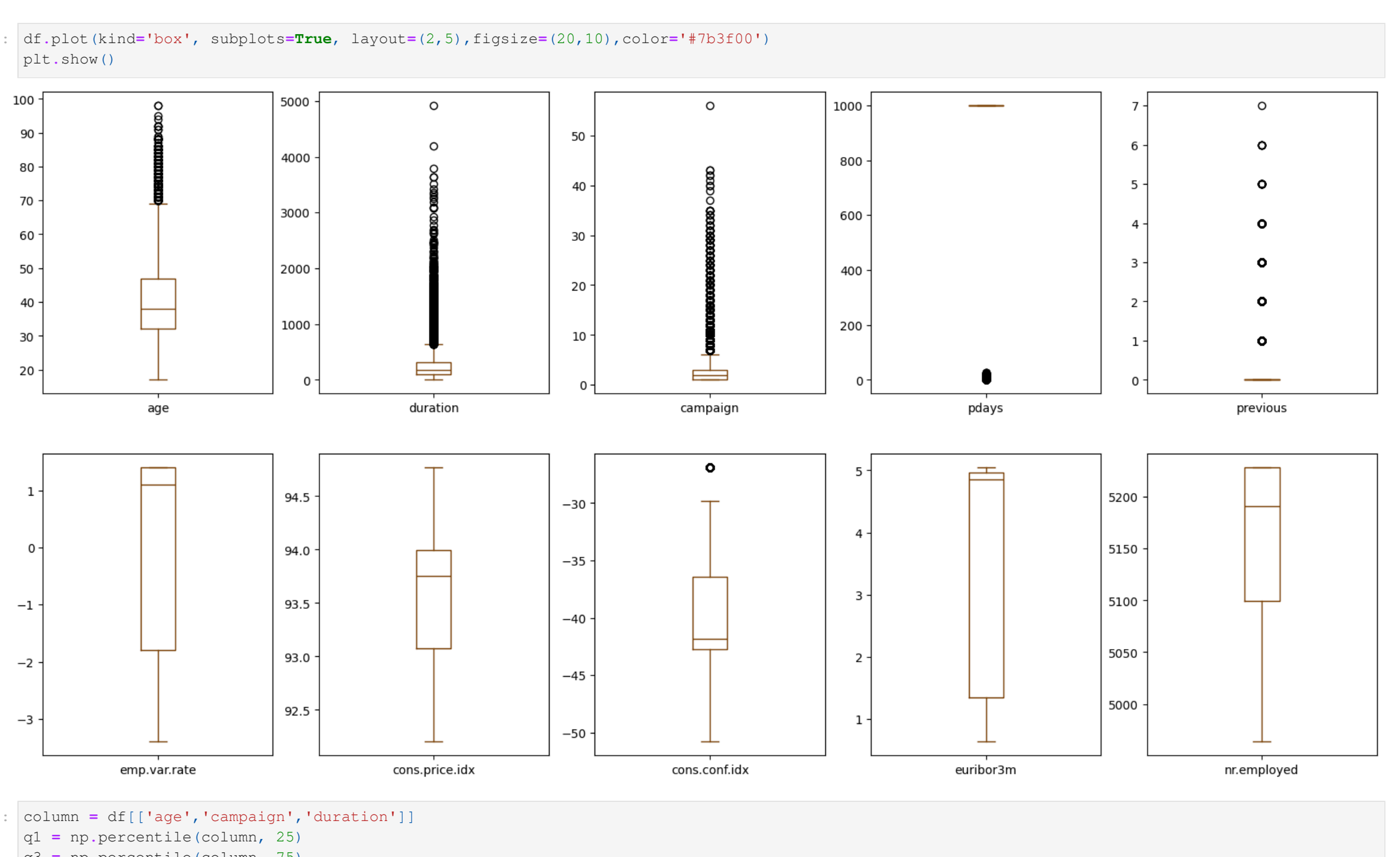
In [18]:
```python
df.hist(figsize=(10,10),color='#00FFFF')
plt.show()
```



In [19]:
```python
# Calculate the number of rows and columns for subplots
num_plots = len(cat_cols)
num_rows = (num_plots + 1) // 2  # Add 1 and divide by 2 to round up for odd numbers
num_cols = 2

# Create a new figure
plt.figure(figsize=(20, 25))  # Adjust the figure size as needed

# Loop through each feature and create a countplot
for i, feature in enumerate(cat_cols, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.countplot(x=feature, data=df, palette='Wistia')
    plt.title(f'Bar Plot of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=45)

# Adjust layout to prevent overlap of subplots
plt.tight_layout()
plt.show()
```

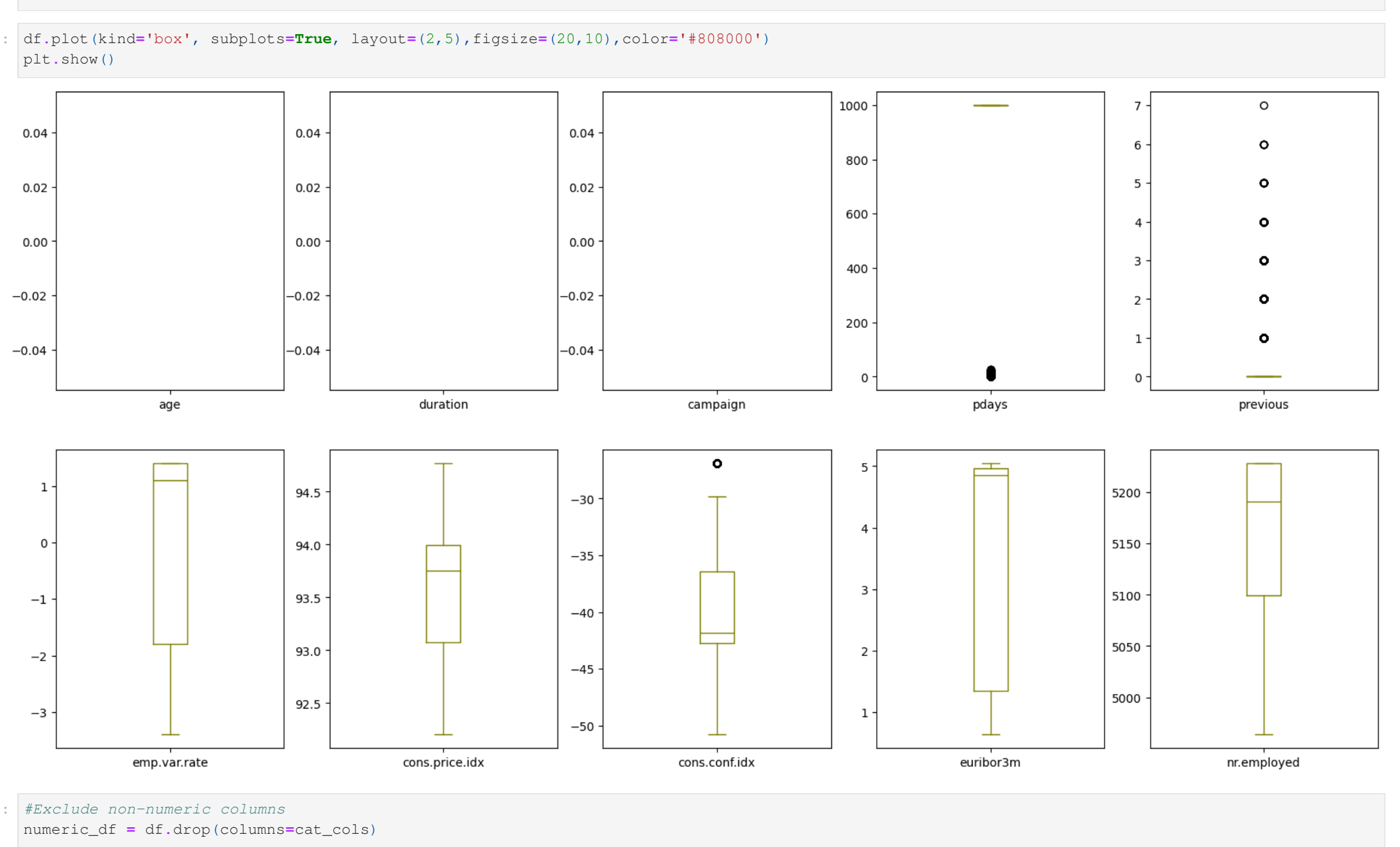


In [20]:
```python
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00')
plt.show()
```



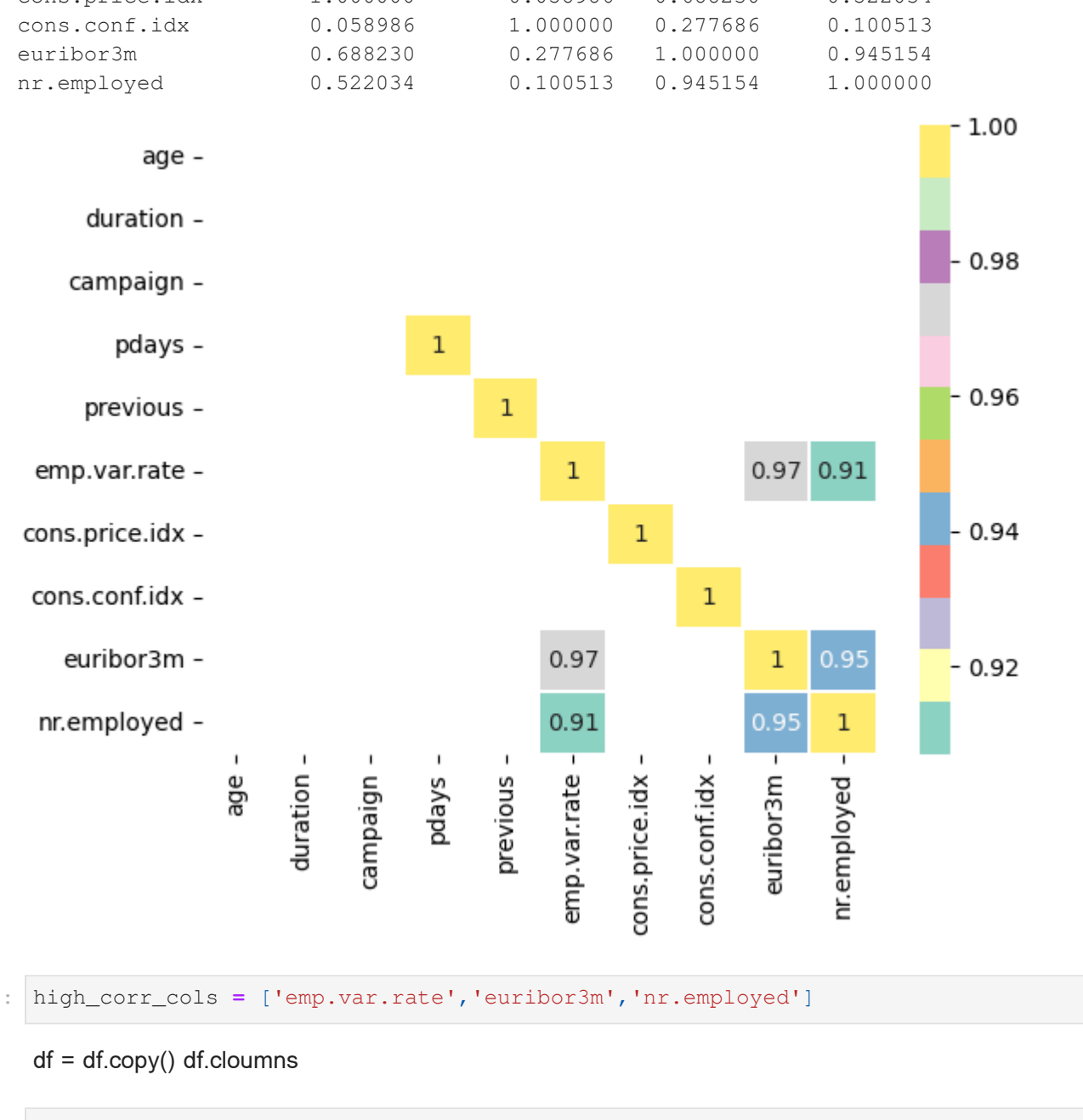
In [22]:
```python
column = df[['age','campaign','duration']]
q1 = np.percentile(column, 25)
q3 = np.percentile(column, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
df[['age','campaign','duration']] = column[(column > lower_bound) & (column < upper_bound)]
```

In [23]:
```python
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#808000')
plt.show()
```



In [24]:
```python
#Exclude non-numeric columns
numeric_df = df.drop(columns=cat_cols)

# Compute the correlation matrix
corr = numeric_df.corr()

# Print the correlation matrix
print(corr)

# Filter correlations with absolute value >= 0.90
corr = corr[abs(corr) >= 0.90]

sns.heatmap(corr,annot=True,cmap='Set3',linewidths=0.2)
plt.show()
```

```
                age  duration  campaign     pdays  previous  emp.var.rate  \
age             NaN       NaN       NaN       NaN       NaN           NaN
duration        NaN       NaN       NaN       NaN       NaN           NaN
campaign        NaN       NaN       NaN       NaN       NaN           NaN
pdays           NaN       NaN       NaN  1.000000 -0.587514      0.271004
previous        NaN       NaN       NaN -0.587514  1.000000     -0.420489
emp.var.rate    NaN       NaN       NaN  0.271004 -0.420489      1.000000
cons.price.idx  NaN       NaN       NaN  0.078889 -0.203130      0.775334
cons.conf.idx   NaN       NaN       NaN -0.091342 -0.050936      0.196041
euribor3m       NaN       NaN       NaN  0.296899 -0.454494      0.972245
nr.employed     NaN       NaN       NaN  0.372605 -0.501333      0.906970

                cons.price.idx  cons.conf.idx  euribor3m  nr.employed
age                        NaN            NaN        NaN          NaN
duration                   NaN            NaN        NaN          NaN
campaign                   NaN            NaN        NaN          NaN
pdays                 0.078889      -0.091342   0.296899     0.372605
previous             -0.203130      -0.050936  -0.454494    -0.501333
emp.var.rate          0.775334       0.196041   0.972245     0.906970
cons.price.idx        1.000000       0.058986   0.688230     0.522034
cons.conf.idx         0.058986       1.000000   0.277686     0.100513
euribor3m             0.688230       0.277686   1.000000     0.945154
nr.employed           0.522034       0.100513   0.945154     1.000000
```



In [29]:
```python
high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

df = df.copy() df.cloumns

In [31]:
```python
df.drop(high_corr_cols,inplace=True,axis=1)  # axis=1 indicates columns
df.columns
```

Out[31]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'cons.price.idx', 'cons.conf.idx', 'deposit'],
      dtype='object')
```

In [32]:
```python
df.shape
```

Out[32]: (41188, 18)

In [53]:
```python
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\santhosh\appdata\local\programs\python\python313\lib\site-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in c:\users\santhosh\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (2.2.5)
Requirement already satisfied: scipy>=1.6.0 in c:\users\santhosh\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in c:\users\santhosh\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\santhosh\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (3.6.0)
[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [56]:
```python
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df.apply(lb.fit_transform)
df_encoded
```

Out[56]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	cons.price.idx	cons.conf.idx	deposit
0	0	3	1	0	0	0	0	1	6	1	0	0	26	0	1	18	16	0
1	0	7	1	3	1	0	0	1	6	1	0	0	26	0	1	18	16	0
2	0	7	1	3	0	2	0	1	6	1	0	0	26	0	1	18	16	0
3	0	0	1	1	0	0	0	1	6	1	0	0	26	0	1	18	16	0
4	0	7	1	3	0	0	2	1	6	1	0	0	26	0	1	18	16	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41183	0	5	1	5	0	2	0	0	7	0	0	0	26	0	1	25	0	1
41184	0	1	1	5	0	0	0	0	7	0	0	0	26	0	1	25	0	0
41185	0	5	1	6	0	2	0	0	7	0	0	0	26	0	1	25	0	0
41186	0	9	1	5	0	0	0	0	7	0	0	0	26	0	1	25	0	1
41187	0	5	1	5	0	2	0	0	7	0	0	0	26	1	0	25	0	0

41188 rows × 18 columns

In [57]:
```python
df_encoded['deposit'].value_counts()
```

Out[57]:
```
deposit
0    36548
1     4640
Name: count, dtype: int64
```

In [58]:
```python
x = df_encoded.drop('deposit',axis=1)  # independent variable
y = df_encoded['deposit']              # dependent variable
print(x.shape)
print(y.shape)
print(type(x))
print(type(y))
```

```
(41188, 17)
(41188,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [61]:
```python
from sklearn.model_selection import train_test_split
```

In [62]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=1)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(30891, 17)
(10297, 17)
(30891,)
(10297,)
```

In [63]:
```python
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

def eval_model(y_test,y_pred):
    acc = accuracy_score(y_test,y_pred)
    print('Accuracy_Score',acc)
    cm = confusion_matrix(y_test,y_pred)
    print('Confusion Matrix\n',cm)
    print('Classification Report\n',classification_report(y_test,y_pred))

def mscore(model):
    train_score = model.score(x_train,y_train)
    test_score = model.score(x_test,y_test)
    print('Training Score',train_score)
    print('Testing Score',test_score)
```

In [64]:
```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='gini',max_depth=5,min_samples_split=10)
dt.fit(x_train,y_train)
```

Out[64]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

In [65]:
```python
mscore(dt)
```

```
Training Score 0.9023340131429867
Testing Score 0.8986112459939788
```

In [66]:
```python
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

```
[0 0 0 ... 0 0 0]
```

In [67]:
```python
eval_model(y_test,ypred_dt)
```

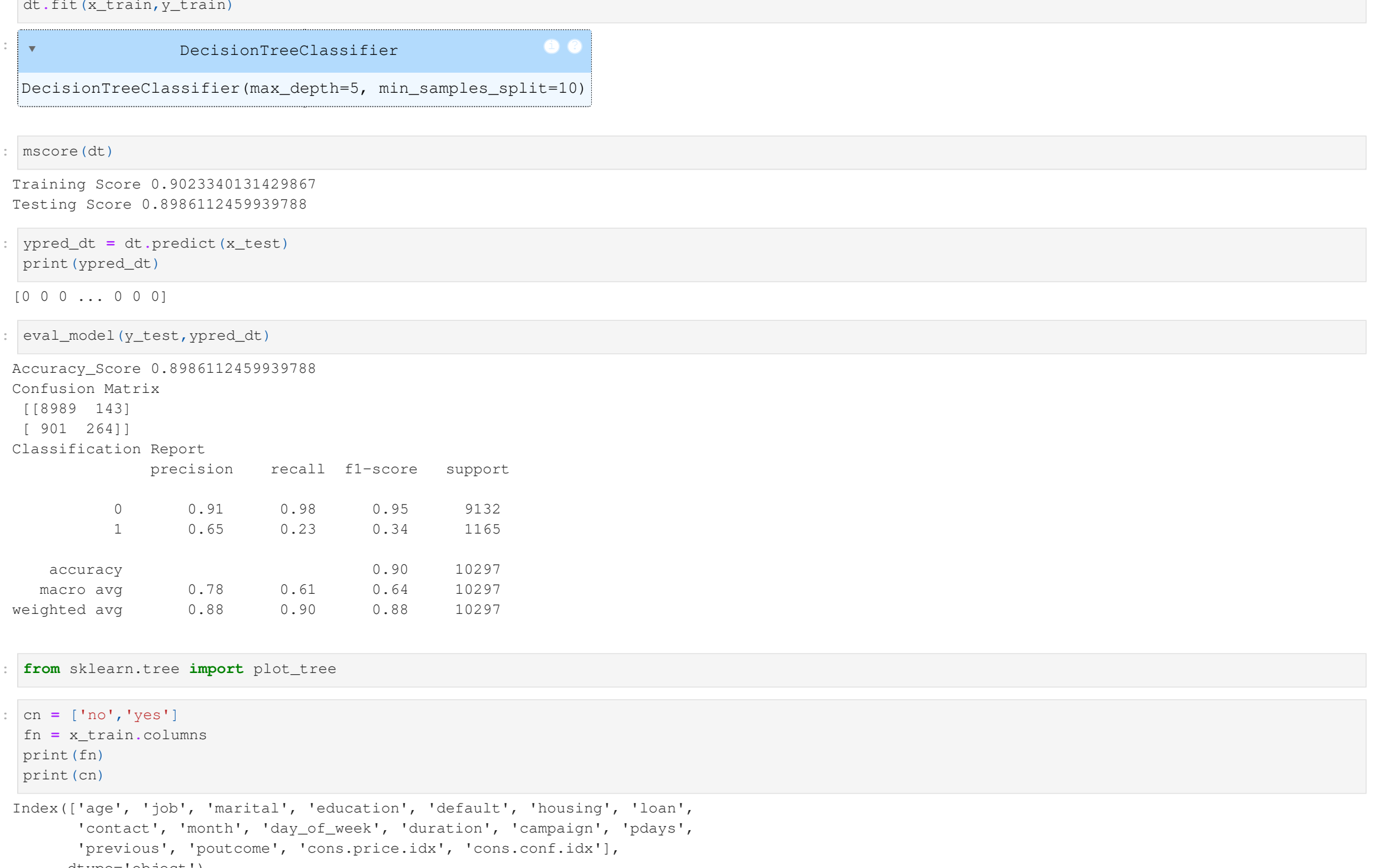
```
Accuracy_Score 0.8986112459939788
Confusion Matrix
 [[8989  143]
 [ 901  264]]
Classification Report
               precision    recall  f1-score   support

           0       0.91      0.98      0.95      9132
           1       0.65      0.23      0.34      1165

    accuracy                           0.90     10297
   macro avg       0.78      0.61      0.64     10297
weighted avg       0.88      0.90      0.88     10297
```

In [68]:
```python
from sklearn.tree import plot_tree
```

In [69]:
```python
cn = ['no','yes']
fn = x_train.columns
print(fn)
print(cn)
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'cons.price.idx', 'cons.conf.idx'],
      dtype='object')
['no', 'yes']
```

In [70]:
```python
plt.figure(figsize=(30,10))
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



In [71]:
```python
dt1 = DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=15)
dt1.fit(x_train,y_train)
```

Out[71]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=15)
```

In [72]:
```python
mscore(dt1)
```

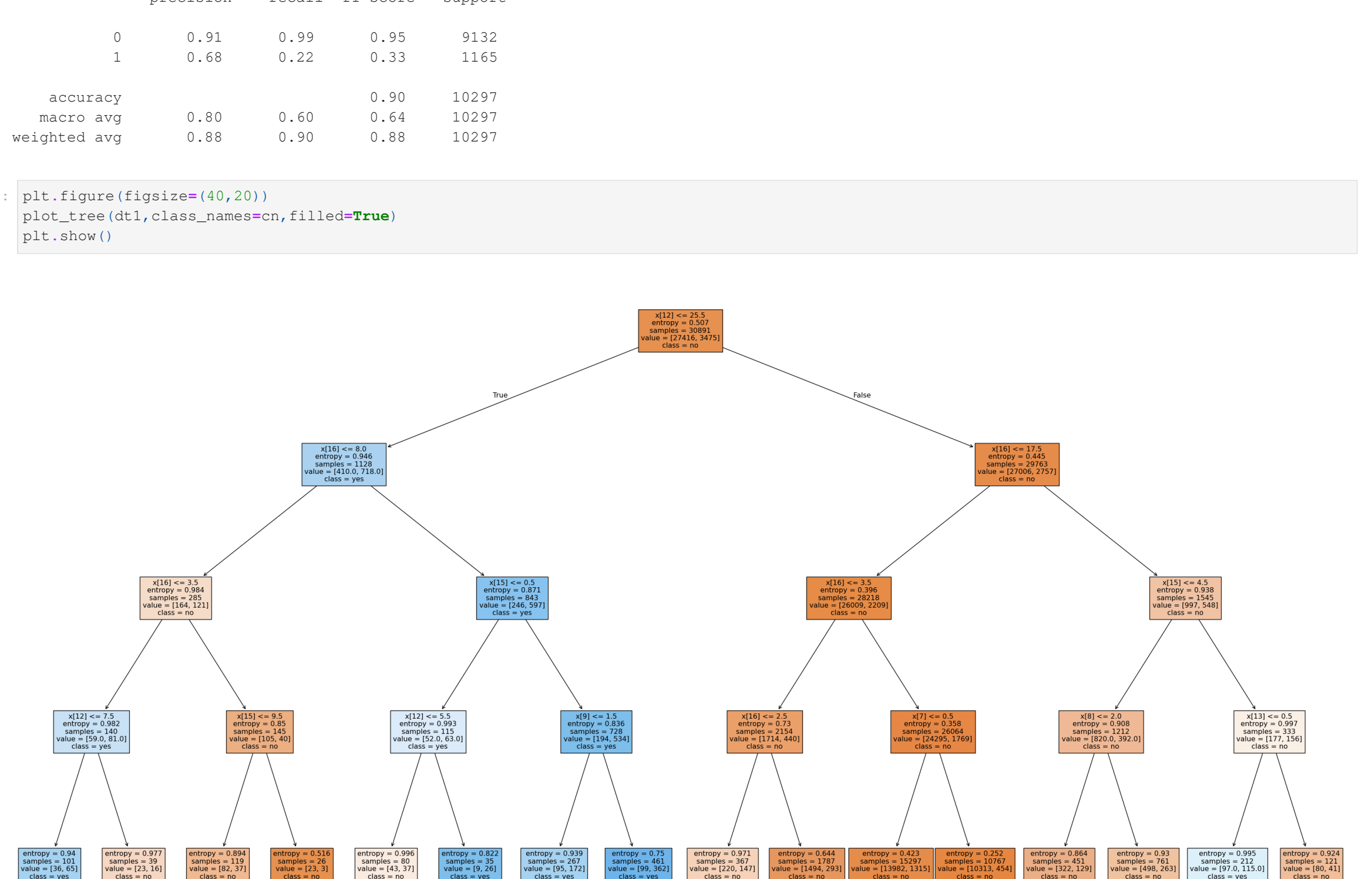
```
Training Score 0.9005859311773656
Testing Score 0.9000679809653297
```

In [73]:
```python
ypred_dt1 = dt1.predict(x_test)
```

In [74]:
```python
eval_model(y_test,ypred_dt1)
```

```
Accuracy_Score 0.9000679809653297
Confusion Matrix
 [[9016  116]
 [ 913  252]]
Classification Report
               precision    recall  f1-score   support

           0       0.91      0.99      0.95      9132
           1       0.68      0.22      0.33      1165

    accuracy                           0.90     10297
   macro avg       0.80      0.60      0.64     10297
weighted avg       0.88      0.90      0.88     10297
```

In [75]:
```python
plt.figure(figsize=(40,20))
plot_tree(dt1,class_names=cn,filled=True)
plt.show()
```



In [ ]: