Bank1

August 17, 2024

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

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

[2]:
```
df = pd.read_csv(r"C:
↪\Users\bobna\Desktop\DS_Task_03\bank-additional\bank-additional\bank-additional.
↪csv",delimiter=';')
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	…	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	deposit
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	…	2	999	0	nonexistent	-1.8	92.893	-46.2	1.313	5099.1	no
1	39	services	single	high.school	no	no	no	telephone	may	fri	…	4	999	0	nonexistent	1.1	93.994	-36.4	4.855	5191.0	no
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	…	1	999	0	nonexistent	1.4	94.465	-41.8	4.962	5228.1	no
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	…	3	999	0	nonexistent	1.4	94.465	-41.8	4.959	5228.1	no
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	…	1	999	0	nonexistent	-0.1	93.200	-42.0	4.191	5195.8	no

[5 rows x 21 columns]

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

[3]:
```
   age          job  marital          education default  housing     loan  \
0   30  blue-collar  married           basic.9y      no      yes       no
1   39     services   single        high.school      no       no       no
2   25     services  married        high.school      no      yes       no
3   38     services  married           basic.9y      no  unknown  unknown
4   47       admin.  married  university.degree      no      yes       no

     contact month day_of_week  …  campaign  pdays  previous     poutcome  \
0   cellular   may         fri  …         2    999         0  nonexistent
1  telephone   may         fri  …         4    999         0  nonexistent
2  telephone   jun         wed  …         1    999         0  nonexistent
3  telephone   jun         fri  …         3    999         0  nonexistent
4   cellular   nov         mon  …         1    999         0  nonexistent

  emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed  deposit
0         -1.8          92.893          -46.2      1.313       5099.1       no
1          1.1          93.994          -36.4      4.855       5191.0       no
2          1.4          94.465          -41.8      4.962       5228.1       no
3          1.4          94.465          -41.8      4.959       5228.1       no
4         -0.1          93.200          -42.0      4.191       5195.8       no

[5 rows x 21 columns]
```

[4]:
```
df.tail()
```

[4]:
```
      age          job  marital    education default housing loan    contact  \
4114   30       admin.  married     basic.6y      no     yes  yes   cellular
4115   39       admin.  married  high.school      no     yes   no  telephone
4116   27      student   single  high.school      no      no   no   cellular
4117   58       admin.  married  high.school      no      no   no   cellular
4118   34   management   single  high.school      no     yes   no   cellular

     month day_of_week  …  campaign  pdays  previous     poutcome  \
4114   jul         thu  …         1    999         0  nonexistent
4115   jul         fri  …         1    999         0  nonexistent
4116   may         mon  …         2    999         1      failure
4117   aug         fri  …         1    999         0  nonexistent
4118   nov         wed  …         1    999         0  nonexistent

     emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed  \
4114          1.4          93.918          -42.7      4.958       5228.1
4115          1.4          93.918          -42.7      4.959       5228.1
4116         -1.8          92.893          -46.2      1.354       5099.1
4117          1.4          93.444          -36.1      4.966       5228.1
4118         -0.1          93.200          -42.0      4.120       5195.8
```

```
      deposit
4114       no
4115       no
4116       no
4117       no
4118       no

[5 rows x 21 columns]
```

[5]:
```
df.dtypes
```

[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
deposit           object
dtype: object
```

[6]:
```
df.dtypes.value_counts()
```

[6]:
```
object     11
int64       5
float64     5
Name: count, dtype: int64
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   age             4119 non-null   int64
 1   job             4119 non-null   object
 2   marital         4119 non-null   object
 3   education       4119 non-null   object
 4   default         4119 non-null   object
 5   housing         4119 non-null   object
 6   loan            4119 non-null   object
 7   contact         4119 non-null   object
 8   month           4119 non-null   object
 9   day_of_week     4119 non-null   object
 10  duration        4119 non-null   int64
 11  campaign        4119 non-null   int64
 12  pdays           4119 non-null   int64
 13  previous        4119 non-null   int64
 14  poutcome        4119 non-null   object
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  deposit         4119 non-null   object
dtypes: float64(5), int64(5), object(11)
memory usage: 675.9+ KB
```

[8]:
```
df.duplicated().sum()
```

[8]:
```
0
```

[9]:
```
df.isna().sum()
```

[9]:
```
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
```

[10]:
```
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')
```

[11]:
```
df.describe()
```

[11]:

	age	duration	campaign	pdays	previous
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337
std	10.313362	254.703736	2.568159	191.922786	0.541788
min	18.000000	0.000000	1.000000	0.000000	0.000000
25%	32.000000	103.000000	1.000000	999.000000	0.000000
50%	38.000000	181.000000	2.000000	999.000000	0.000000
75%	47.000000	317.000000	3.000000	999.000000	0.000000
max	88.000000	3643.000000	35.000000	999.000000	6.000000

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	0.084972	93.579704	-40.499102	3.621356	5166.481695
std	1.563114	0.579349	4.594578	1.733591	73.667904
min	-3.400000	92.201000	-50.800000	0.635000	4963.600000
25%	-1.800000	93.075000	-42.700000	1.334000	5099.100000
50%	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	1.400000	94.767000	-26.900000	5.045000	5228.100000

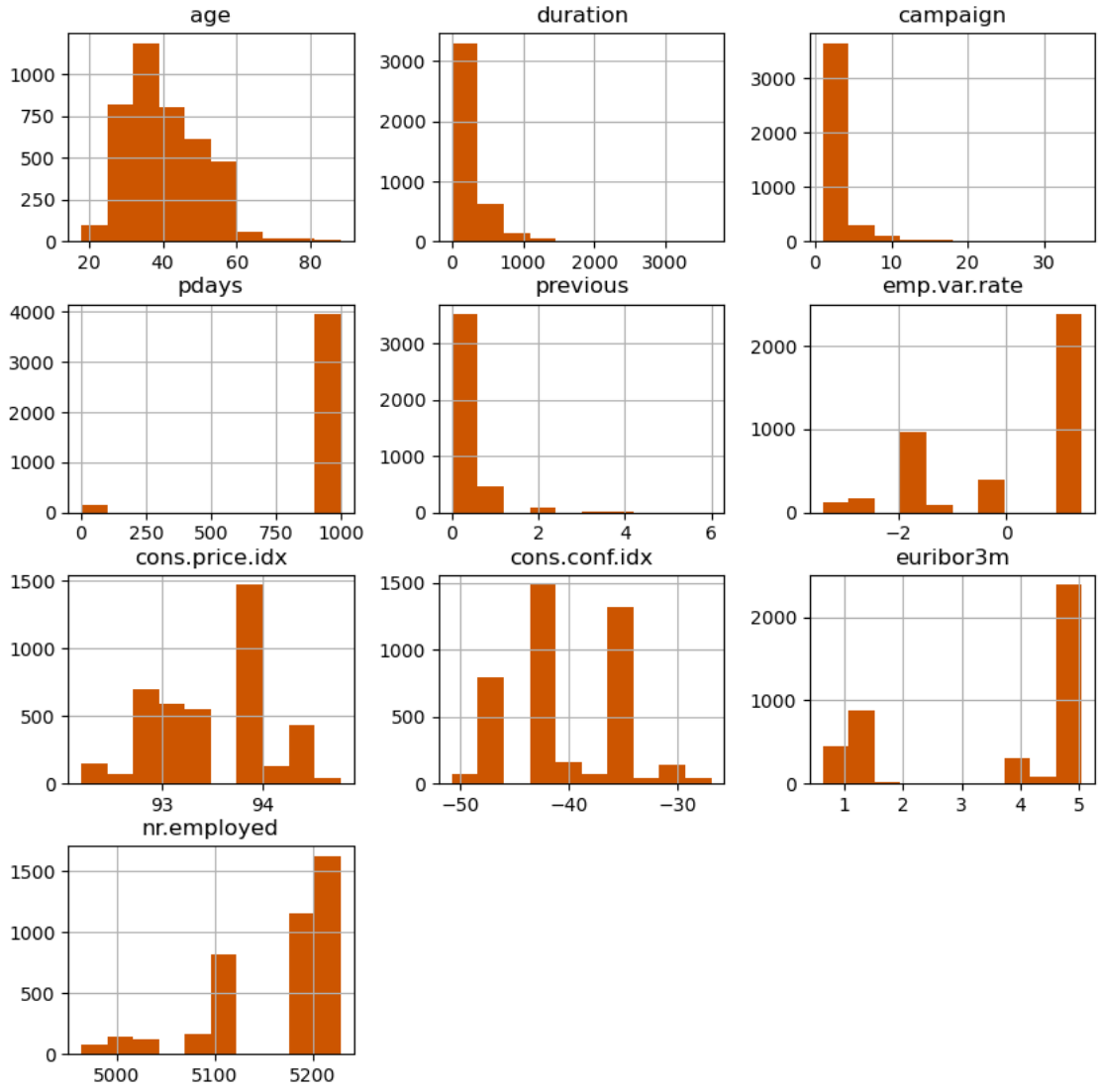
[12]:
```
df.describe(include='object')
```

[12]:

	job	marital	education	default	housing	loan	contact
count	4119	4119	4119	4119	4119	4119	4119
unique	12	4	8	3	3	3	2
top	admin.	married	university.degree	no	yes	no	cellular
freq	1012	2509	1264	3315	2175	3349	2652

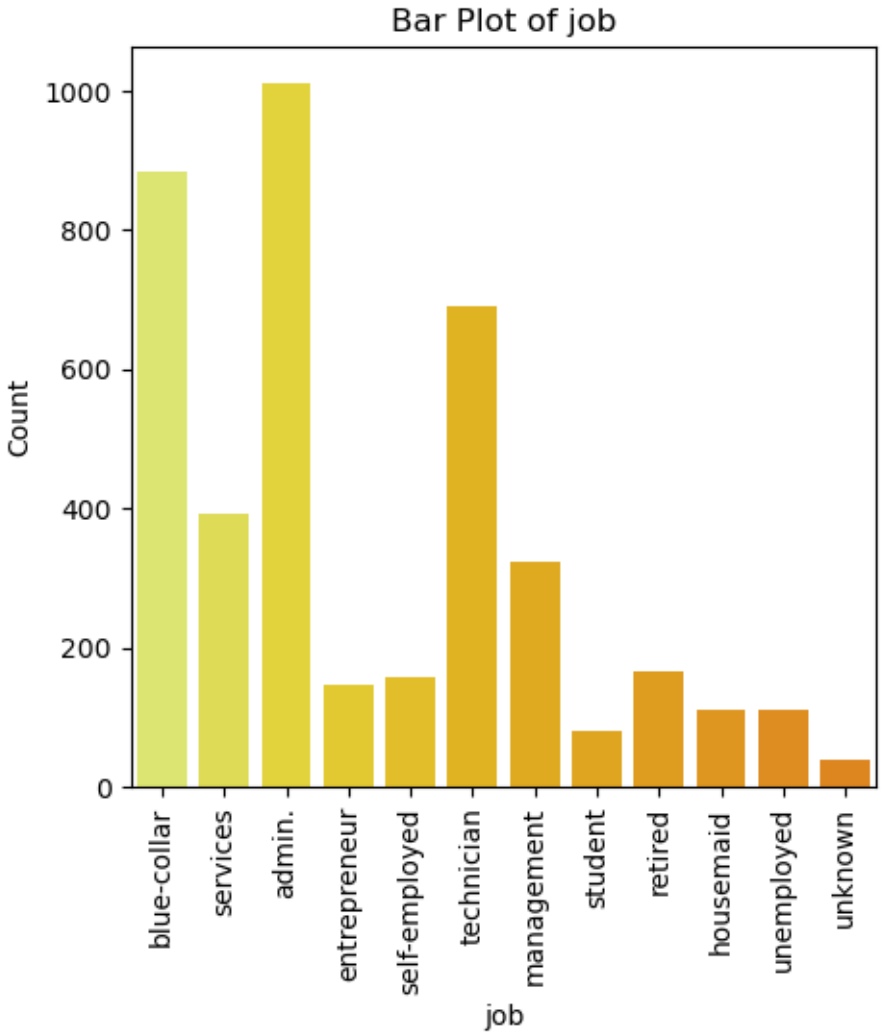
```
        month day_of_week     poutcome deposit
count    4119        4119         4119    4119
unique     10           5            3       2
top       may         thu  nonexistent      no
freq     1378         860         3523    3668
```

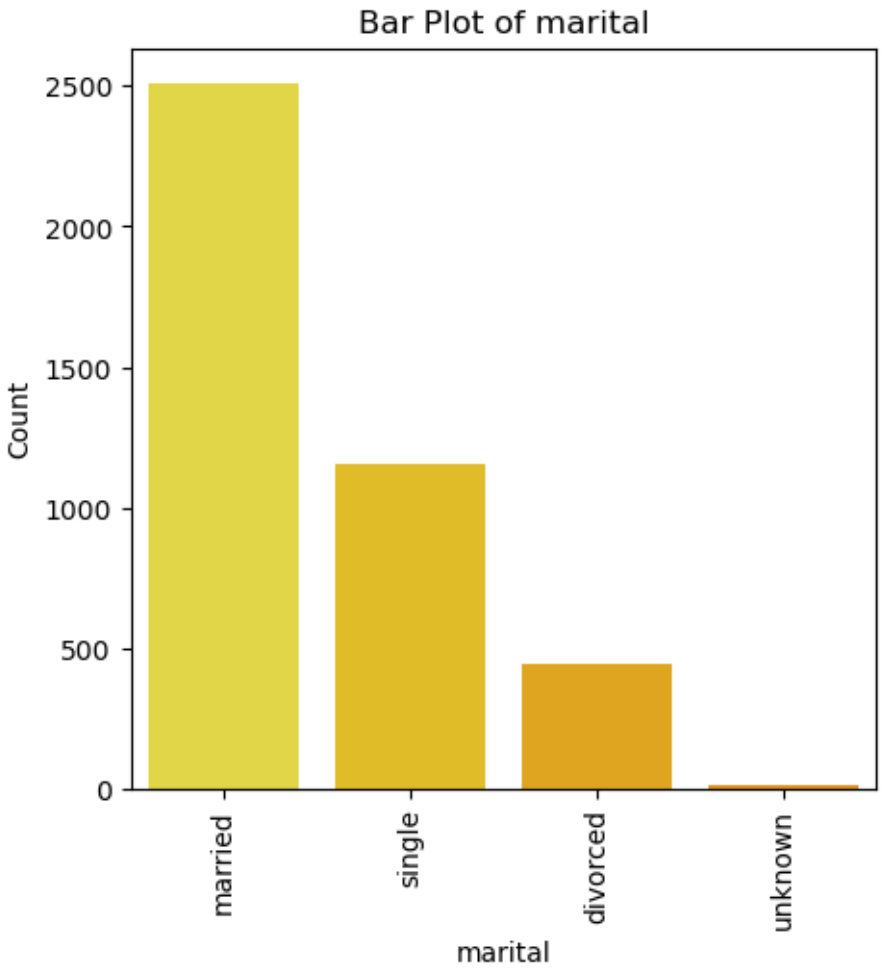
[15]:
```
df.hist(figsize=(10,10),color='#cc5500')
plt.show()
```

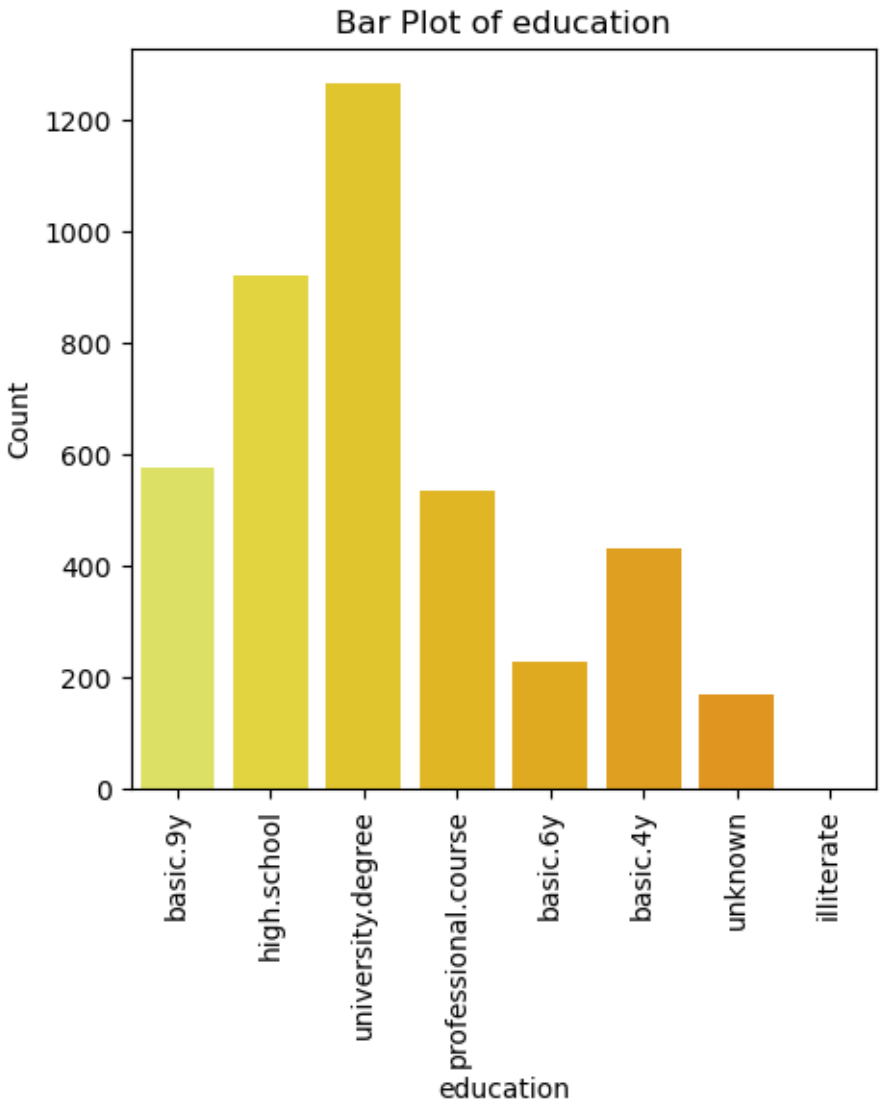


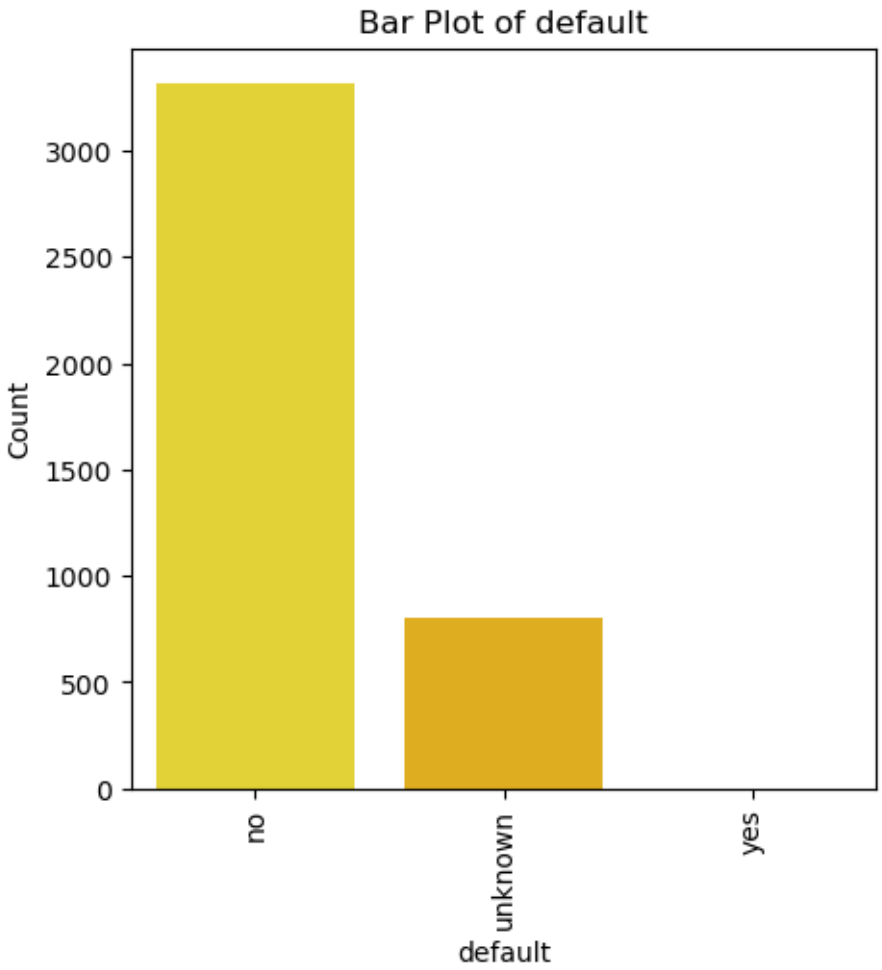
[16]:
```
for feature in cat_cols:
    plt.figure(figsize=(5,5))  # Adjust the figure size as needed
    sns.countplot(x=feature, data=df, palette='Wistia')
```

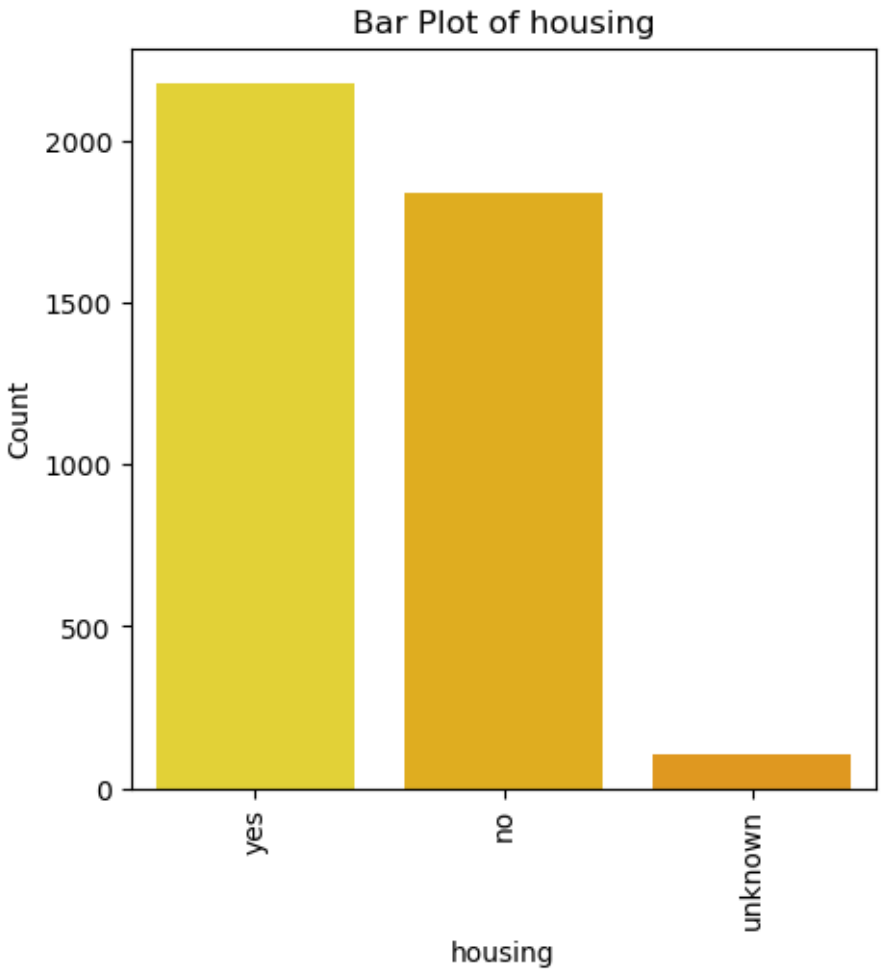
```
    plt.title(f'Bar Plot of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=90)
    plt.show()
```

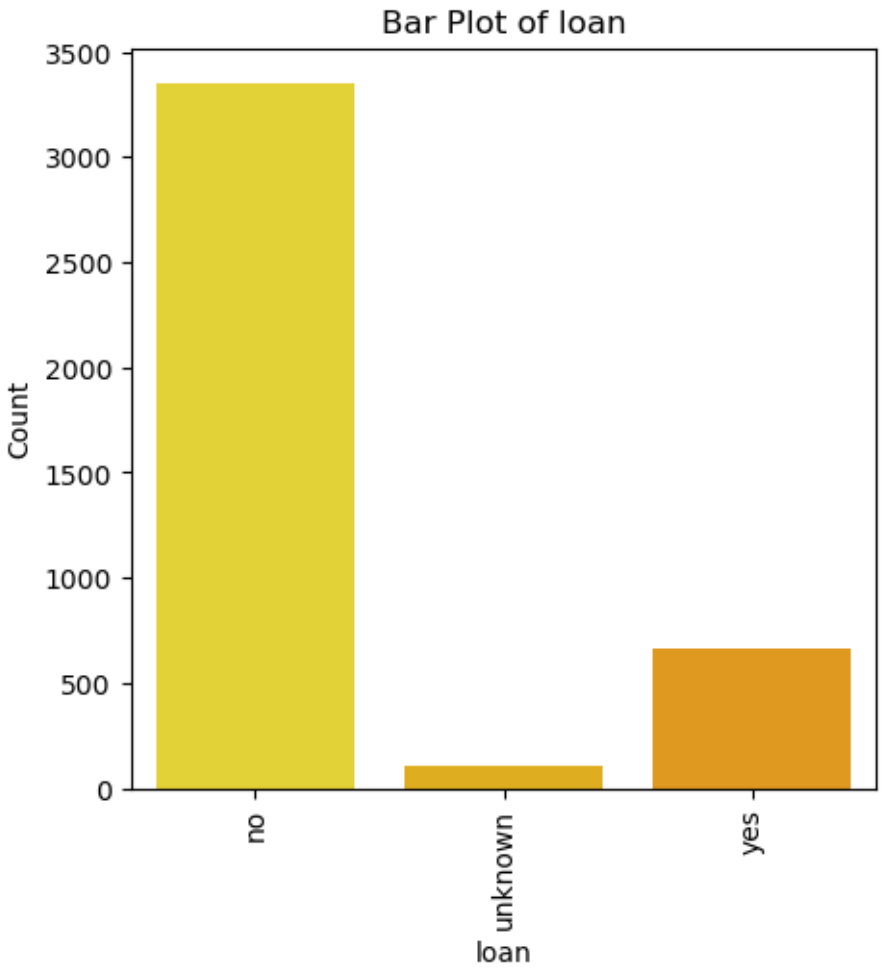


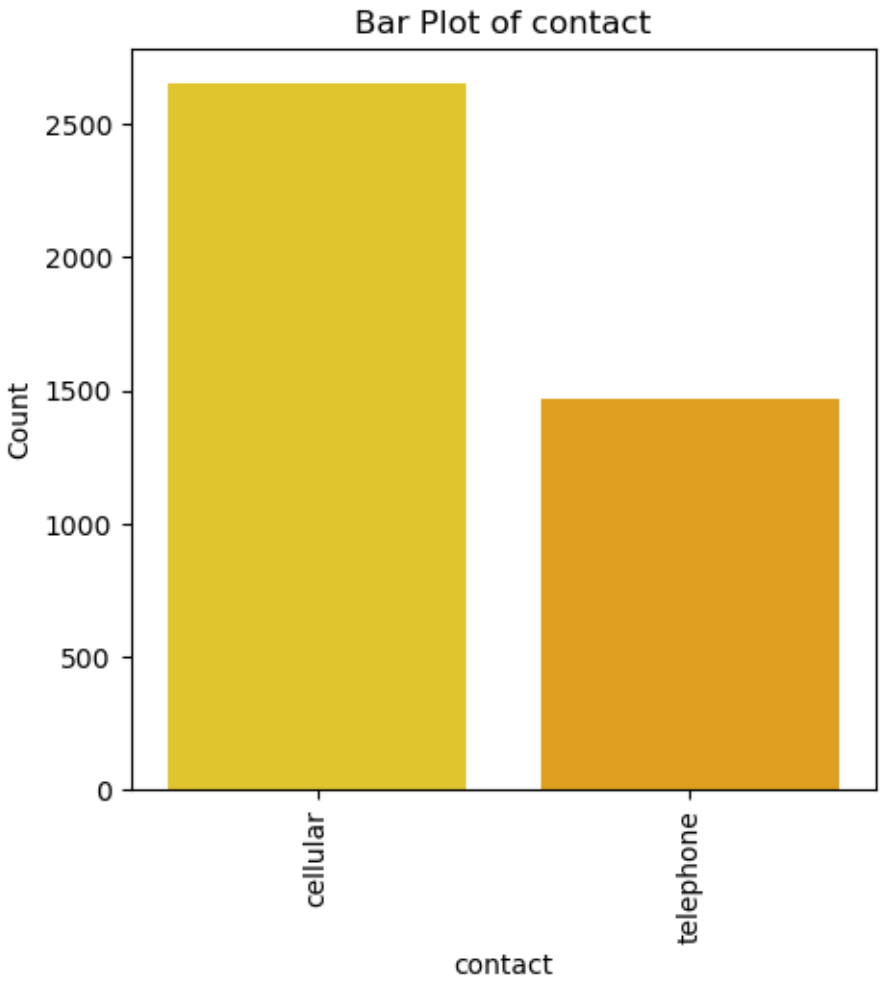

Bar Plot of marital
Count
2500
2000
1500
1000
500
0
married
single
divorced
unknown
marital

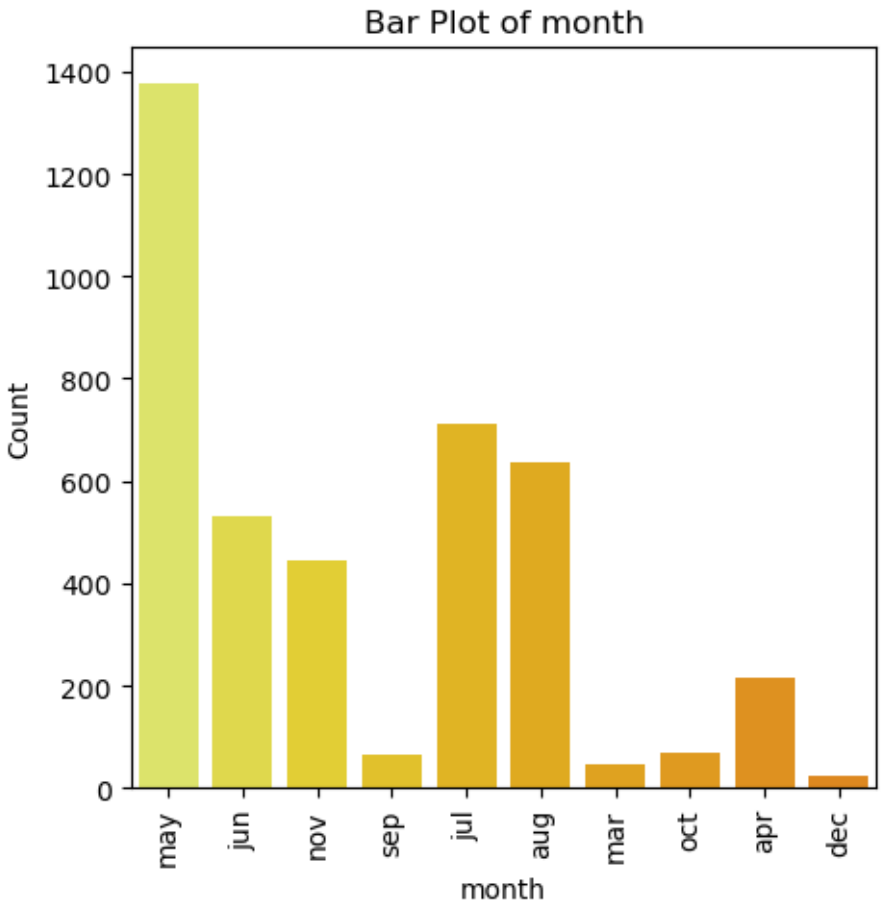

Bar Plot of education
Count
0
200
400
600
800
1000
1200
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate
education

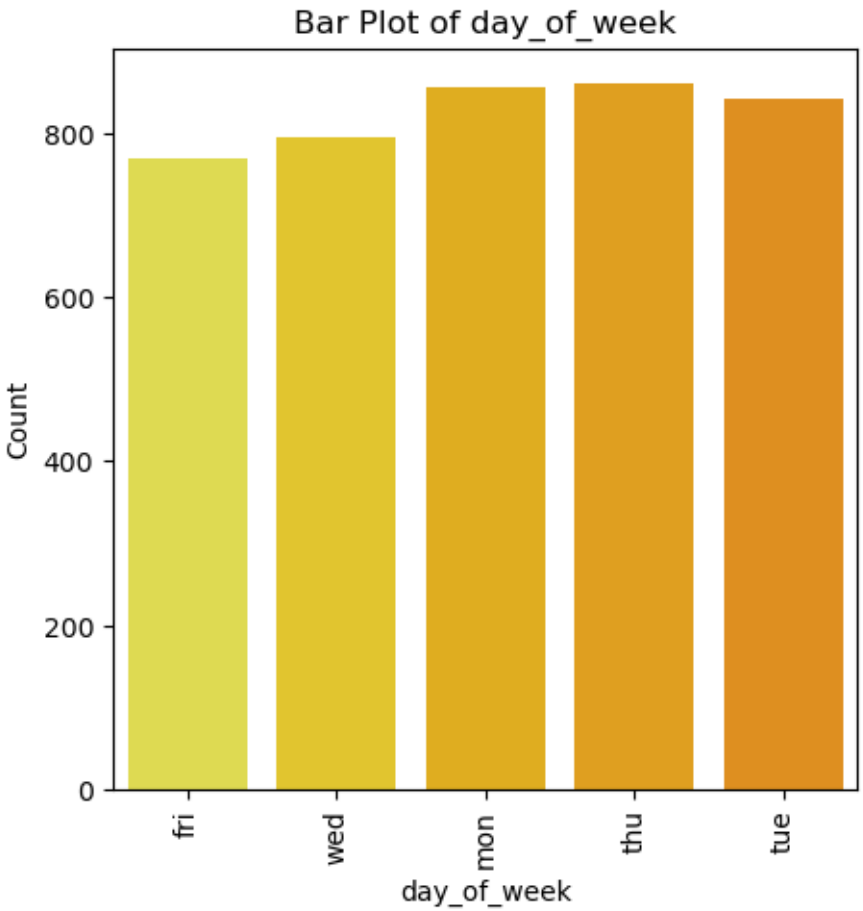

Bar Plot of default
Count
3000
2500
2000
1500
1000
500
0
no
unknown
yes
default

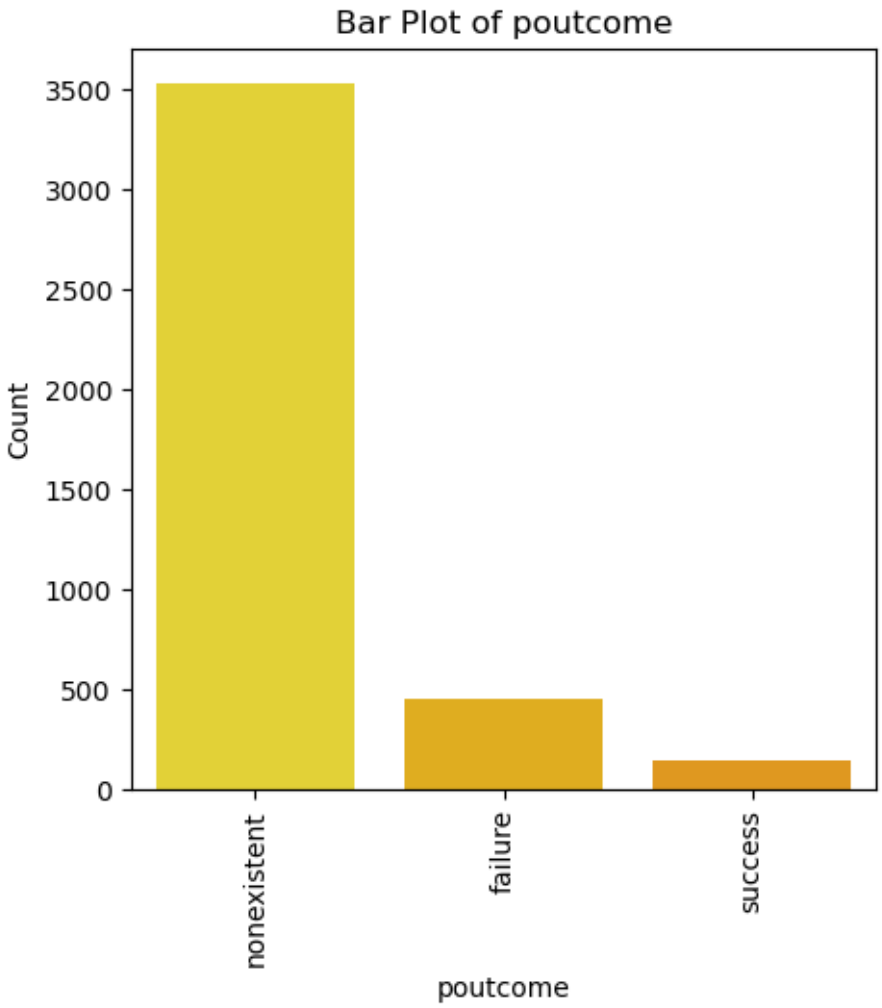

Bar Plot of housing
2000
1500
1000
500
0
Count
yes
no
unknown
housing

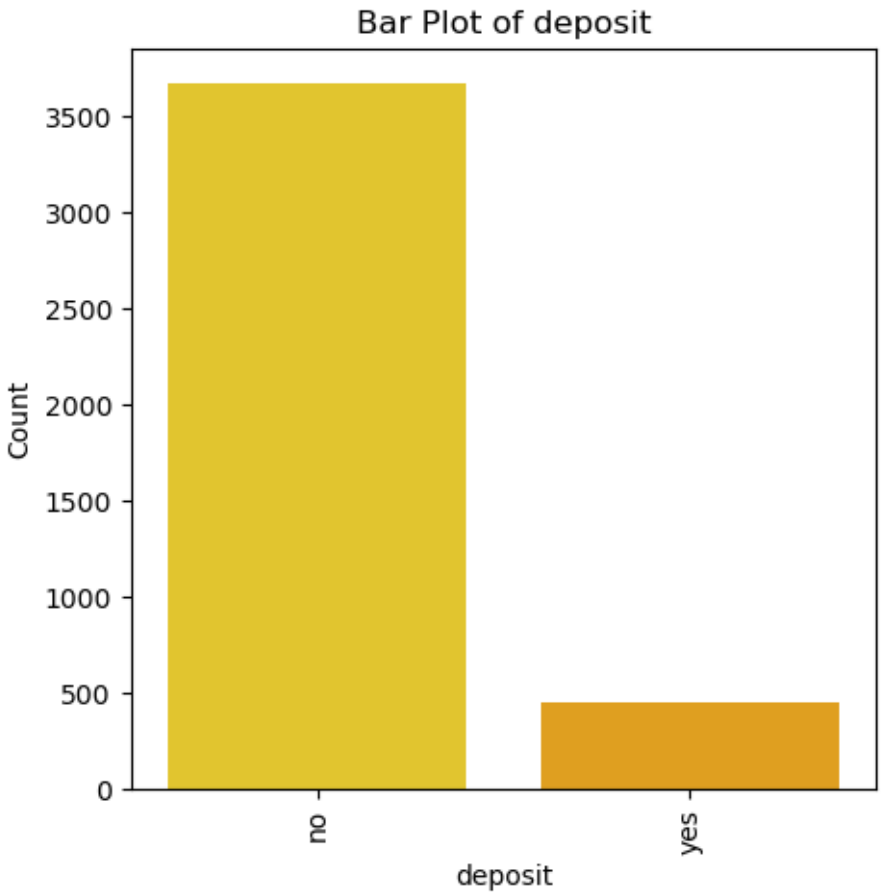

Bar Plot of loan
3500
3000
2500
2000
1500
1000
500
0
Count
no
unknown
yes
loan


Bar Plot of contact
Count
2500
2000
1500
1000
500
0
cellular
telephone
contact

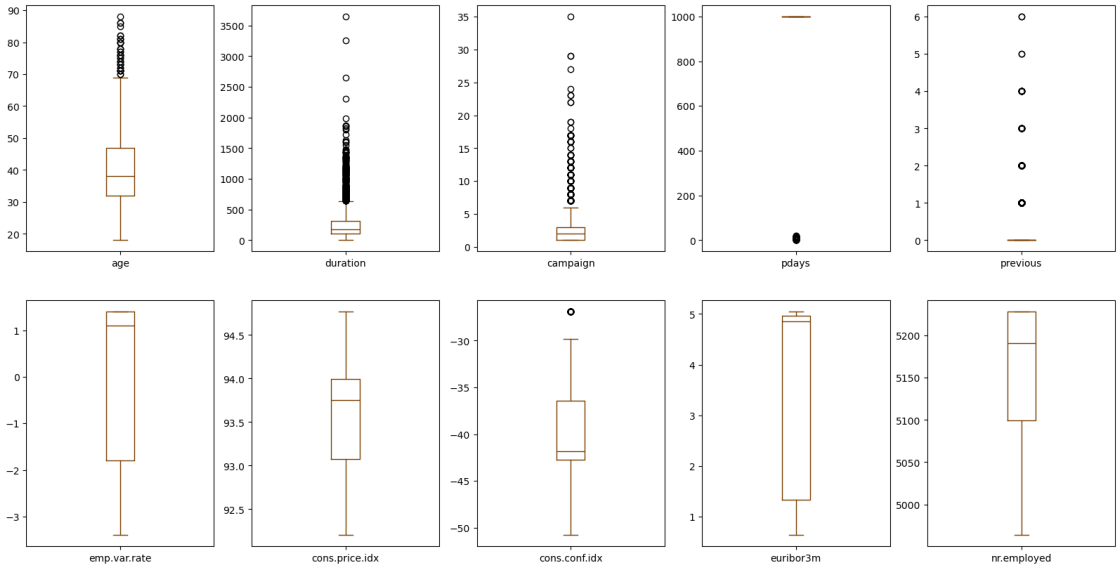

Bar Plot of month
Count
1400
1200
1000
800
600
400
200
0
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
month


Bar Plot of day_of_week
Count
0
200
400
600
800
fri
wed
mon
thu
tue
day_of_week


Bar Plot of poutcome
Count
3500
3000
2500
2000
1500
1000
500
0
nonexistent
failure
success
poutcome

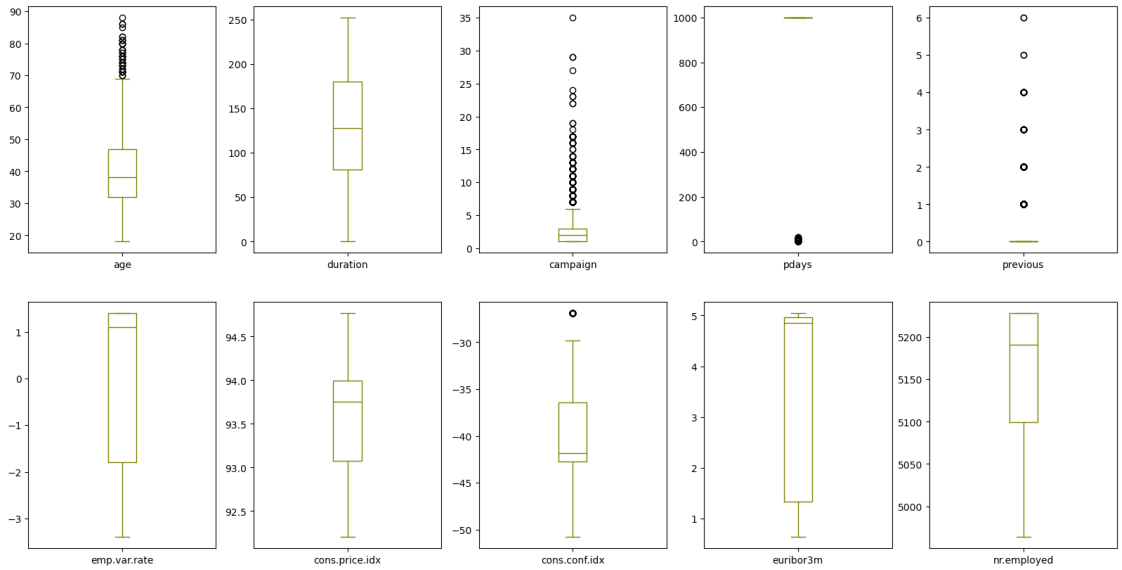


[17]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00')
plt.show()
```



[18]:
```
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)]
```

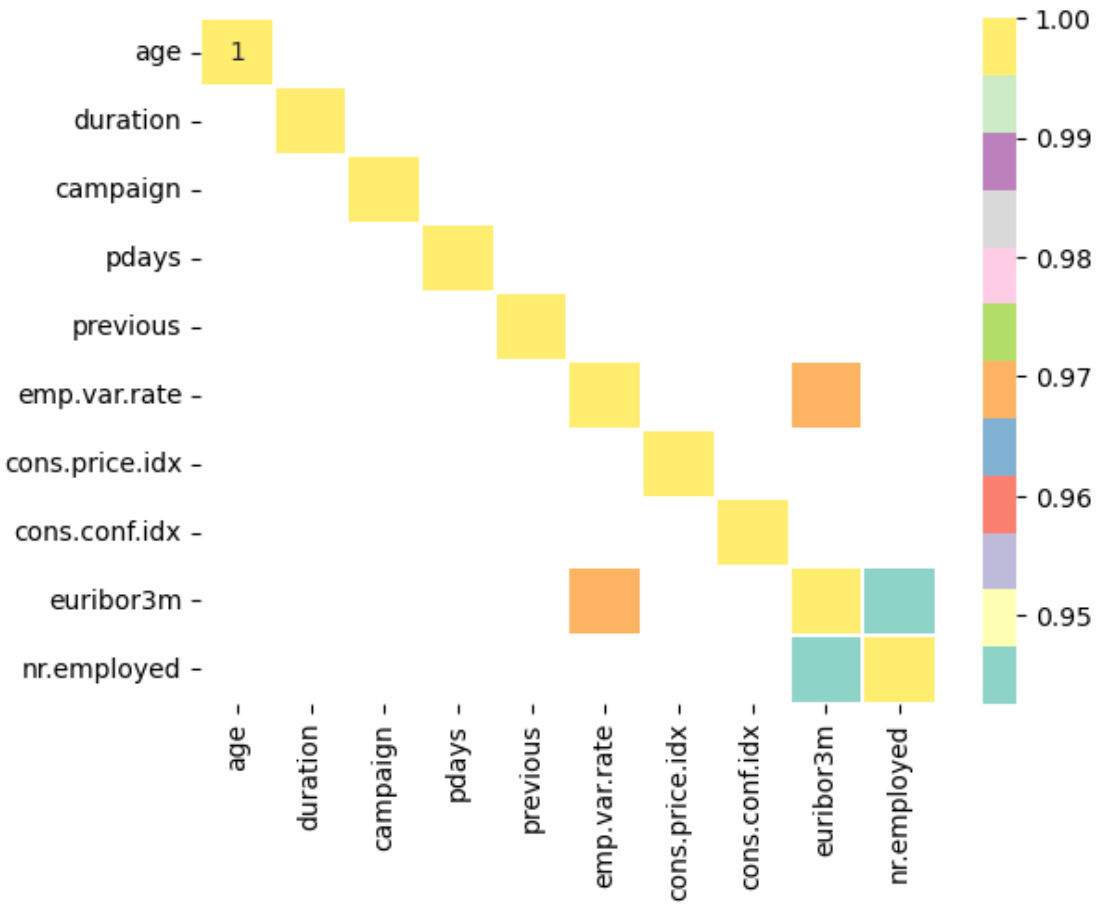
[19]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#808000')
plt.show()
```



[23]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select only numerical columns
numeric_df = df.select_dtypes(include=[float, int])

# Calculate correlation matrix
corr = numeric_df.corr()

# Filter correlation matrix based on threshold (keeping only correlations >= 0.
↪90)
corr_filtered = corr[abs(corr) >= 0.90]
```

```
# Print the filtered correlation matrix
print(corr_filtered)

# Plot the heatmap of the filtered correlation matrix
sns.heatmap(corr_filtered, annot=True, cmap='Set3', linewidths=0.2)
plt.show()
```

```
                age  duration  campaign  pdays  previous  emp.var.rate  \
age             1.0       NaN       NaN    NaN       NaN           NaN
duration        NaN       1.0       NaN    NaN       NaN           NaN
campaign        NaN       NaN       1.0    NaN       NaN           NaN
pdays           NaN       NaN       NaN    1.0       NaN           NaN
previous        NaN       NaN       NaN    NaN       1.0           NaN
emp.var.rate    NaN       NaN       NaN    NaN       NaN      1.000000
cons.price.idx  NaN       NaN       NaN    NaN       NaN           NaN
cons.conf.idx   NaN       NaN       NaN    NaN       NaN           NaN
euribor3m       NaN       NaN       NaN    NaN       NaN      0.970308
nr.employed     NaN       NaN       NaN    NaN       NaN           NaN

                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                      NaN            NaN        NaN          NaN
previous                   NaN            NaN        NaN          NaN
emp.var.rate               NaN            NaN   0.970308          NaN
cons.price.idx             1.0            NaN        NaN          NaN
cons.conf.idx              NaN            1.0        NaN          NaN
euribor3m                  NaN            NaN   1.000000     0.942589
nr.employed                NaN            NaN   0.942589     1.000000
```



[24]:
```
high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

[25]:
```
df1 = df.copy()
df1.columns
```

[25]:
```
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')
```

[26]:
```
df1.drop(high_corr_cols,inplace=True,axis=1)  # axis=1 indicates columns
df1.columns
```

[26]:
```
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')
```

[27]:
```
df1.shape
```

[27]:
```
(4119, 18)
```

[28]:
```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

[28]:

	age	job	marital	education	default	housing	loan	contact	month	\
0	12	1	1	2	0	2	0	0	6	
1	21	7	2	3	0	0	0	1	6	
2	7	7	1	3	0	2	0	1	4	
3	20	7	1	2	0	1	1	1	4	
4	29	0	1	6	0	2	0	0	7	
…	…	…	…	…	…	…	…	…	…	
4114	12	0	1	1	0	2	2	0	3	
4115	21	0	1	3	0	2	0	1	3	
4116	9	8	2	3	0	0	0	0	6	
4117	40	0	1	3	0	0	0	0	1	
4118	16	4	2	3	0	2	0	0	7	

	day_of_week	duration	campaign	pdays	previous	poutcome	\
0	0	250	1	20	0	1	
1	0	250	3	20	0	1	
2	4	224	0	20	0	1	
3	0	14	2	20	0	1	
4	1	55	0	20	0	1	
…	…	…	…	…	…	…	
4114	2	50	0	20	0	1	
4115	0	216	0	20	0	1	
4116	1	61	1	20	1	0	
4117	0	250	0	20	0	1	
4118	4	172	0	20	0	1	

	cons.price.idx	cons.conf.idx	deposit
0	8	4	0
1	18	16	0
2	23	8	0
3	23	8	0
4	11	7	0
…	…	…	…
4114	17	6	0
4115	17	6	0
4116	8	4	0
4117	13	17	0
4118	11	7	0

```
[4119 rows x 18 columns]
```

[29]:
```
df_encoded['deposit'].value_counts()
```

[29]:
```
deposit
0    3668
1     451
Name: count, dtype: int64
```

[30]:
```
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))
```

```
(4119, 17)
(4119,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

[31]:
```
from sklearn.model_selection import train_test_split

print(4119*0.25)
```

```
1029.75
```

[32]:
```
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)
```

```
(3089, 17)
(1030, 17)
(3089,)
(1030,)
```

[33]:
```
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)
```

[34]:
```
from sklearn.tree import DecisionTreeClassifier

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

[34]:
```
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

[35]:
```
mscore(dt)
```

```
Training Score 0.9148591777274199
Testing Score 0.8990291262135922
```

[36]:
```
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

```
[0 0 1 … 0 0 0]
```

[37]:
```
eval_model(y_test,ypred_dt)
```

```
Accuracy_Score 0.8990291262135922
Confusion Matrix
 [[905  25]
 [ 79  21]]
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       930
           1       0.46      0.21      0.29       100

    accuracy                           0.90      1030
   macro avg       0.69      0.59      0.62      1030
weighted avg       0.87      0.90      0.88      1030
```

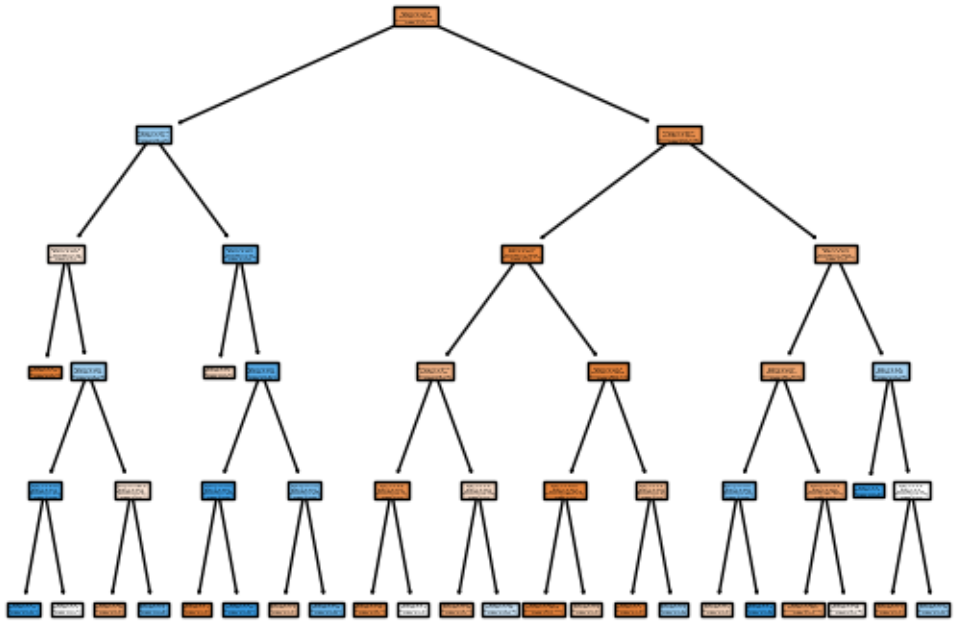
[38]:
```
from sklearn.tree import plot_tree
```

[39]:
```
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']
```

[40]:
```
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



[41]:
```
dt1 =␣
  ↪DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=15)
dt1.fit(x_train,y_train)
```

[41]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=15)
```

[42]:
```
mscore(dt1)
```

```
Training Score 0.9080608611201036
Testing Score 0.9048543689320389
```

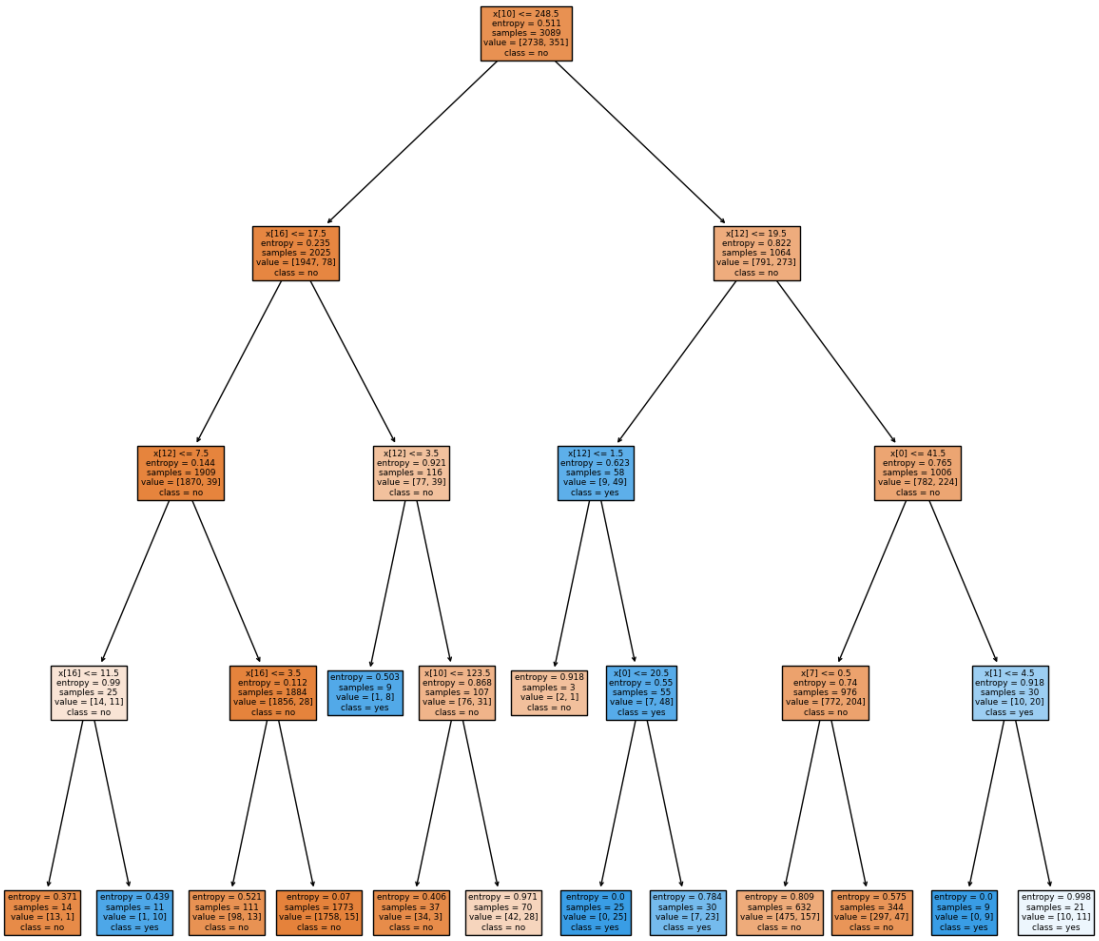
[43]:
```
ypred_dt1 = dt1.predict(x_test)
```

[44]:
```
eval_model(y_test,ypred_dt1)
```

```
Accuracy_Score 0.9048543689320389
Confusion Matrix
 [[915  15]
```

```
 [ 83  17]]
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       930
           1       0.53      0.17      0.26       100

    accuracy                           0.90      1030
   macro avg       0.72      0.58      0.60      1030
weighted avg       0.88      0.90      0.88      1030
```

[45]:
```
plt.figure(figsize=(15,15))
plot_tree(dt1,class_names=cn,filled=True)
plt.show()
```



[ ]: