

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

import warnings
warnings.filterwarnings('ignore')

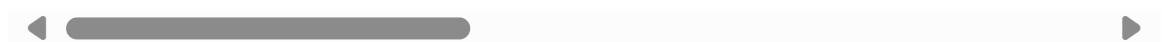
%matplotlib inline
```

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

```
Out[2]:
```

	age	job	marital	education	default	housing	loan	contact	month
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	ma
1	39	services	single	high.school	no	no	no	telephone	ma
2	25	services	married	high.school	no	yes	no	telephone	ju
3	38	services	married	basic.9y	no	unknown	unknown	telephone	ju
4	47	admin.	married	university.degree	no	yes	no	cellular	no

5 rows × 21 columns



```
In [3]: df.tail()
```

```
Out[3]:
```

	age	job	marital	education	default	housing	loan	contact	month
4114	30	admin.	married	basic.6y	no	yes	yes	cellular	ju
4115	39	admin.	married	high.school	no	yes	no	telephone	ju
4116	27	student	single	high.school	no	no	no	cellular	may
4117	58	admin.	married	high.school	no	no	no	cellular	aug
4118	34	management	single	high.school	no	yes	no	cellular	nov

5 rows × 21 columns



```
In [4]: df.shape
```

```
Out[4]: (4119, 21)
```

```
In [5]: df.columns
```

```
Out[5]: 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 [6]: df.dtypes
```

```
Out[6]: 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 [7]: df.dtypes.value_counts()
```

```
Out[7]: object      11  
int64         5  
float64       5  
Name: count, dtype: int64
```

```
In [8]: 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
```

```
In [9]: df.duplicated().sum()
```

```
Out[9]: 0
```

```
In [10]: df.isna().sum()
```

```
Out[10]: 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 [11]: 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 [12]: `df.describe()`

Out[12]:

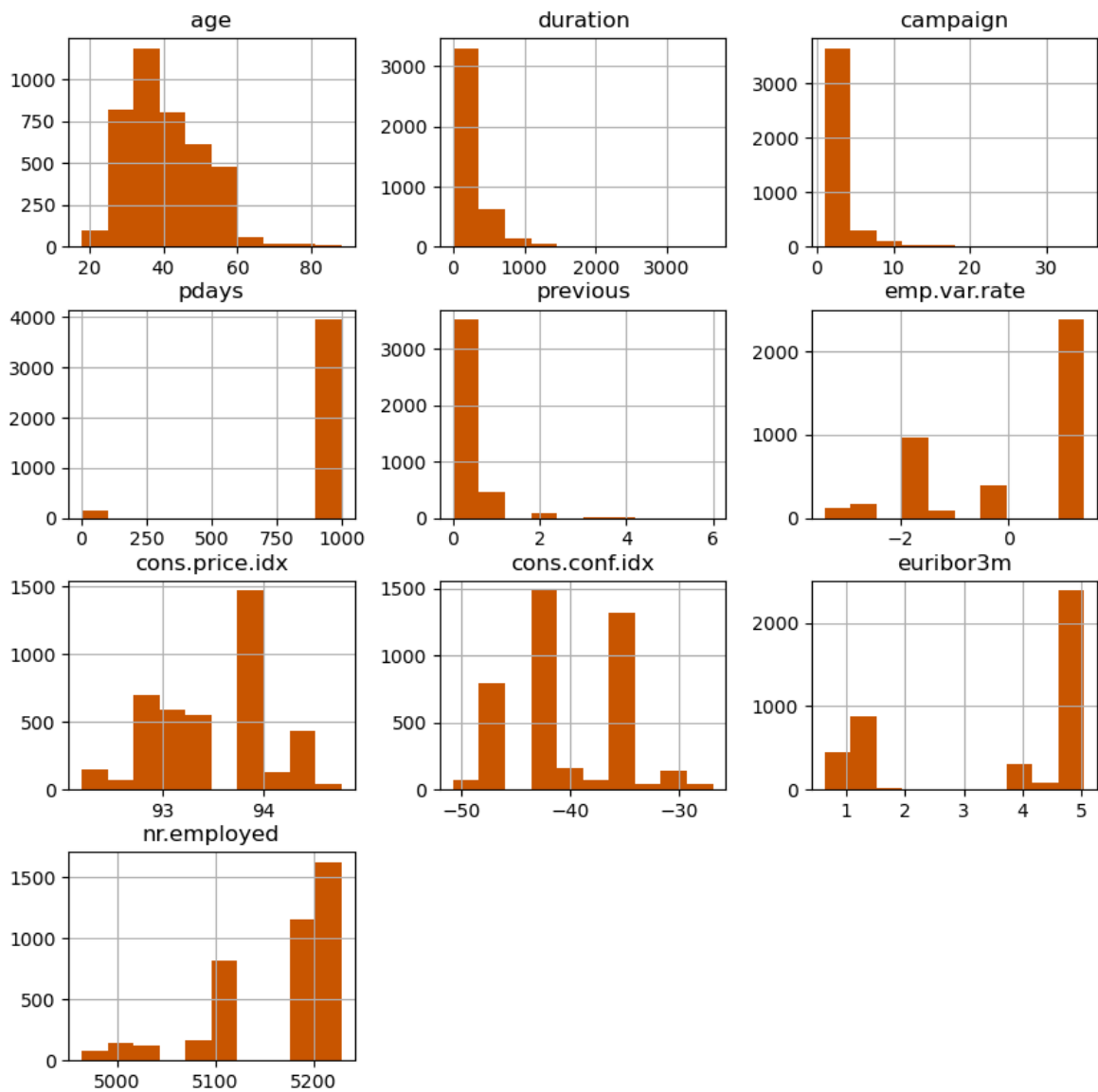
	age	duration	campaign	pdays	previous	emp.var.rate
<b>count</b>	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
<b>mean</b>	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972
<b>std</b>	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114
<b>min</b>	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000
<b>25%</b>	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000
<b>50%</b>	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000
<b>75%</b>	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000
<b>max</b>	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000

In [13]: `df.describe(include='object')`

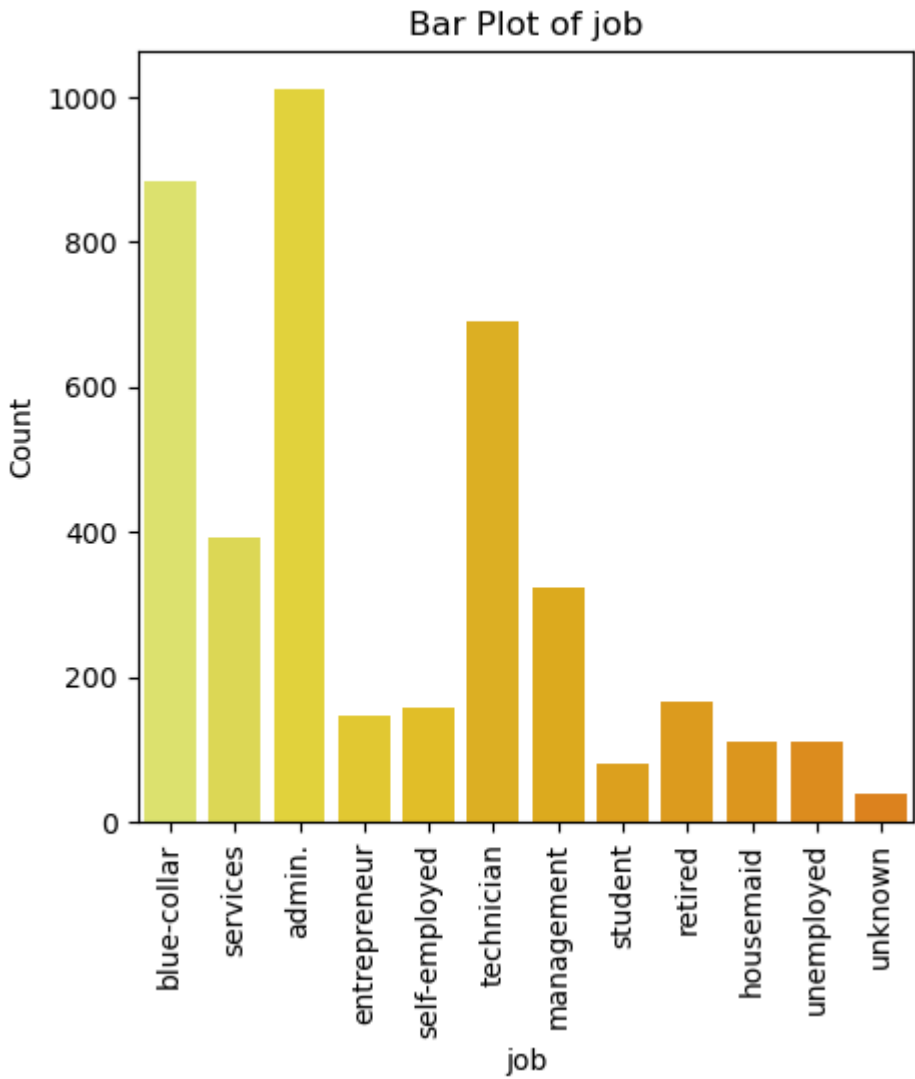
Out[13]:

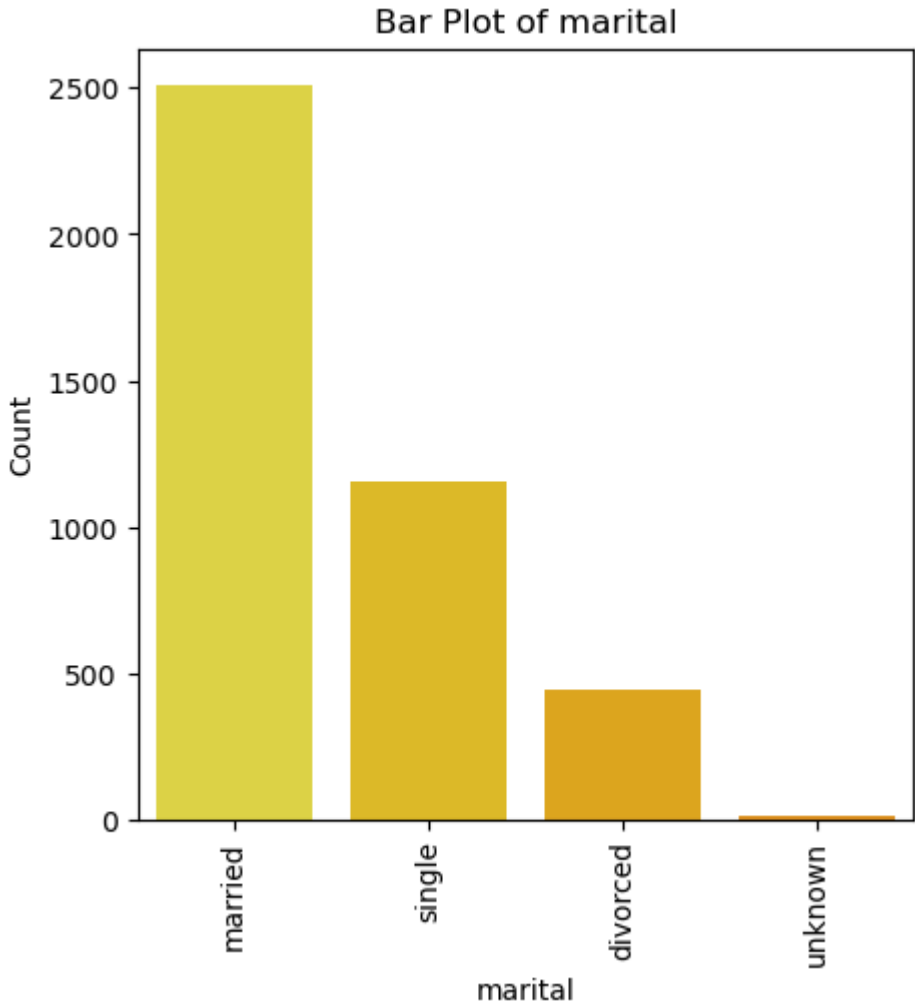
	job	marital	education	default	housing	loan	contact	month	day
<b>count</b>	4119	4119	4119	4119	4119	4119	4119	4119	4119
<b>unique</b>	12	4	8	3	3	3	2	10	10
<b>top</b>	admin.	married	university.degree	no	yes	no	cellular	may	mon
<b>freq</b>	1012	2509	1264	3315	2175	3349	2652	1378	1378

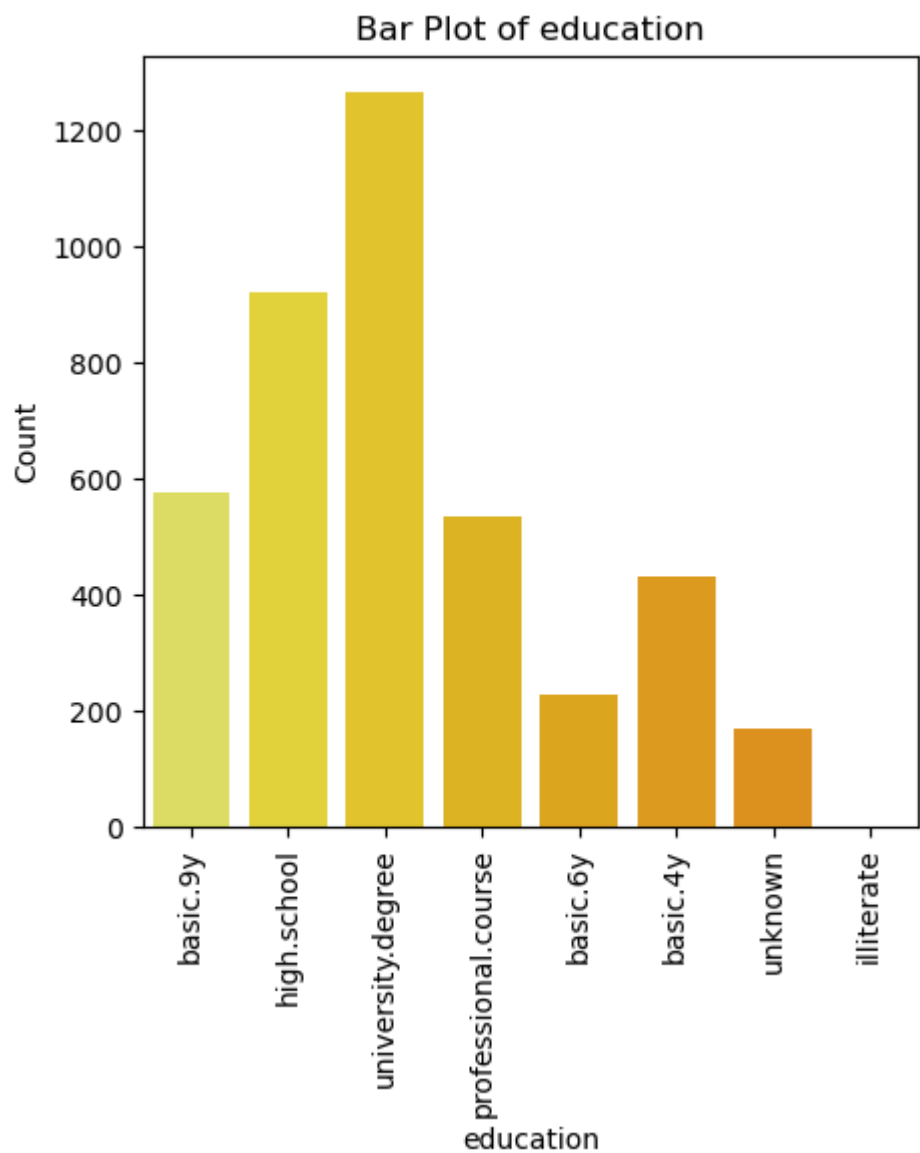
In [14]: `df.hist(figsize=(10,10),color='#cc5500')`  
`plt.show()`



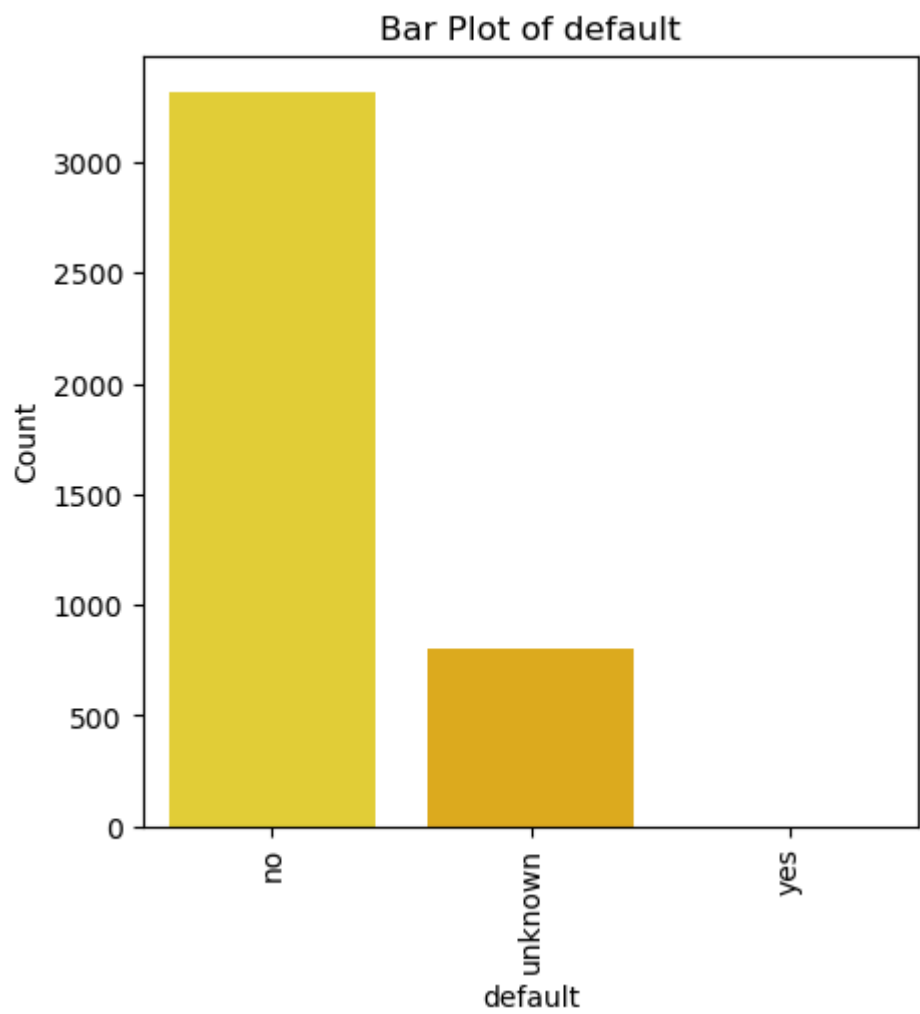
```
In [15]: for feature in cat_cols:
plt.figure(figsize=(5,5)) # Adjust the figure size as needed
sns.countplot(x=feature, data=df, palette='Wistia')
plt.title(f'Bar Plot of {feature}')
plt.xlabel(feature)
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```

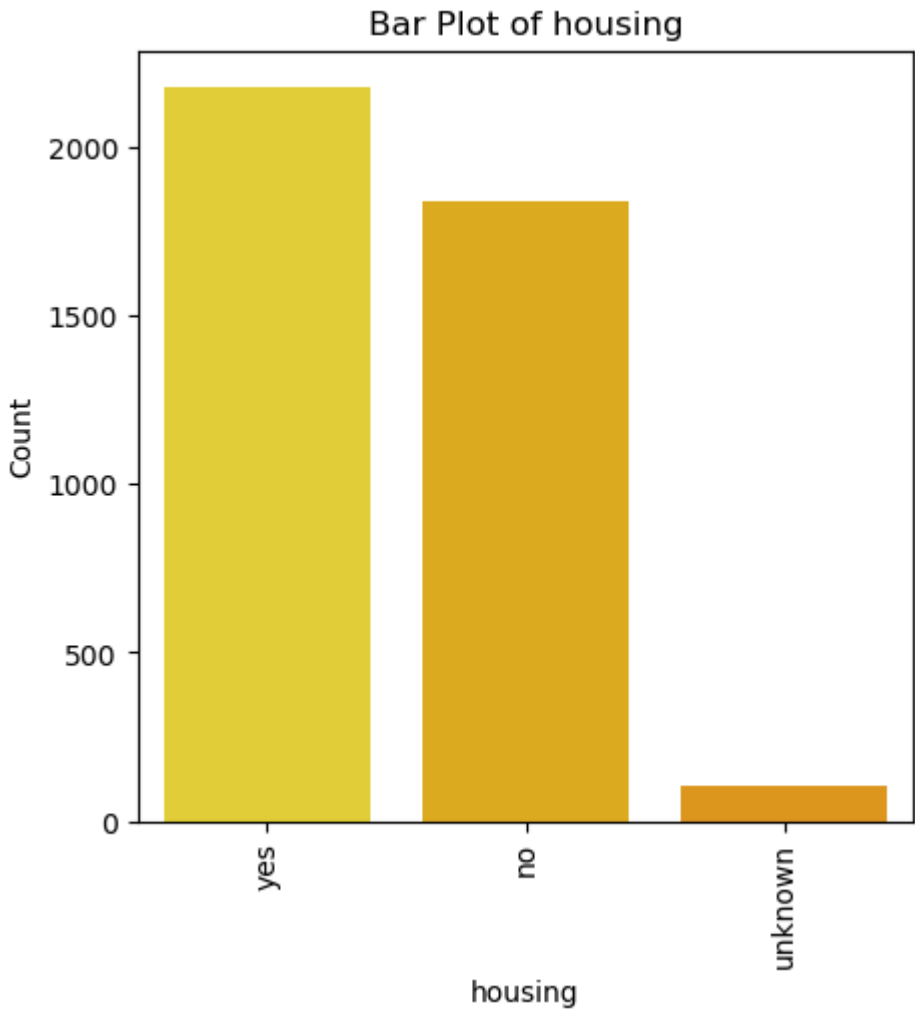


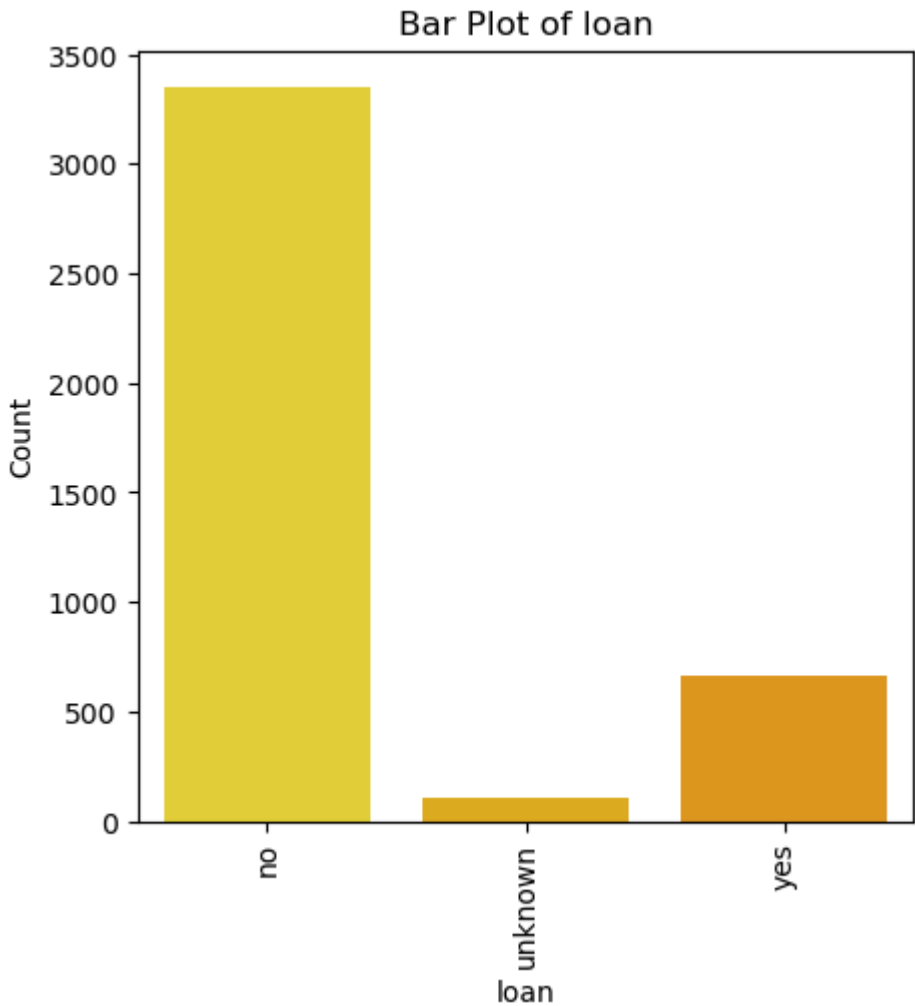


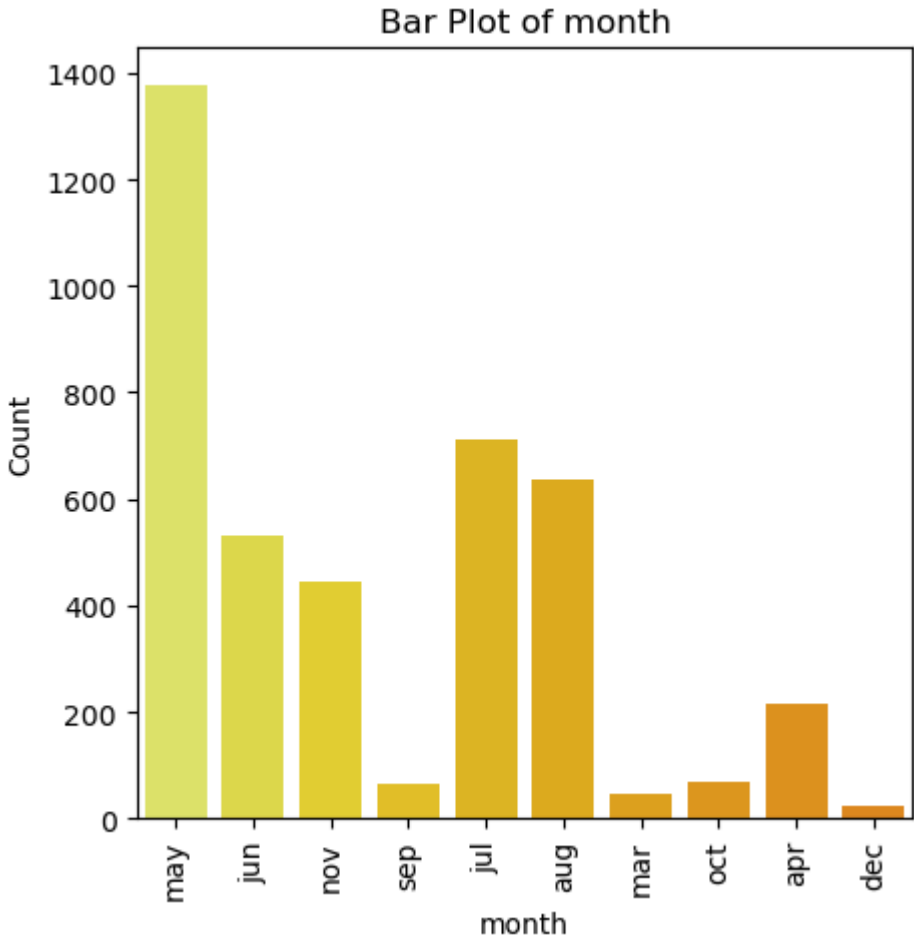
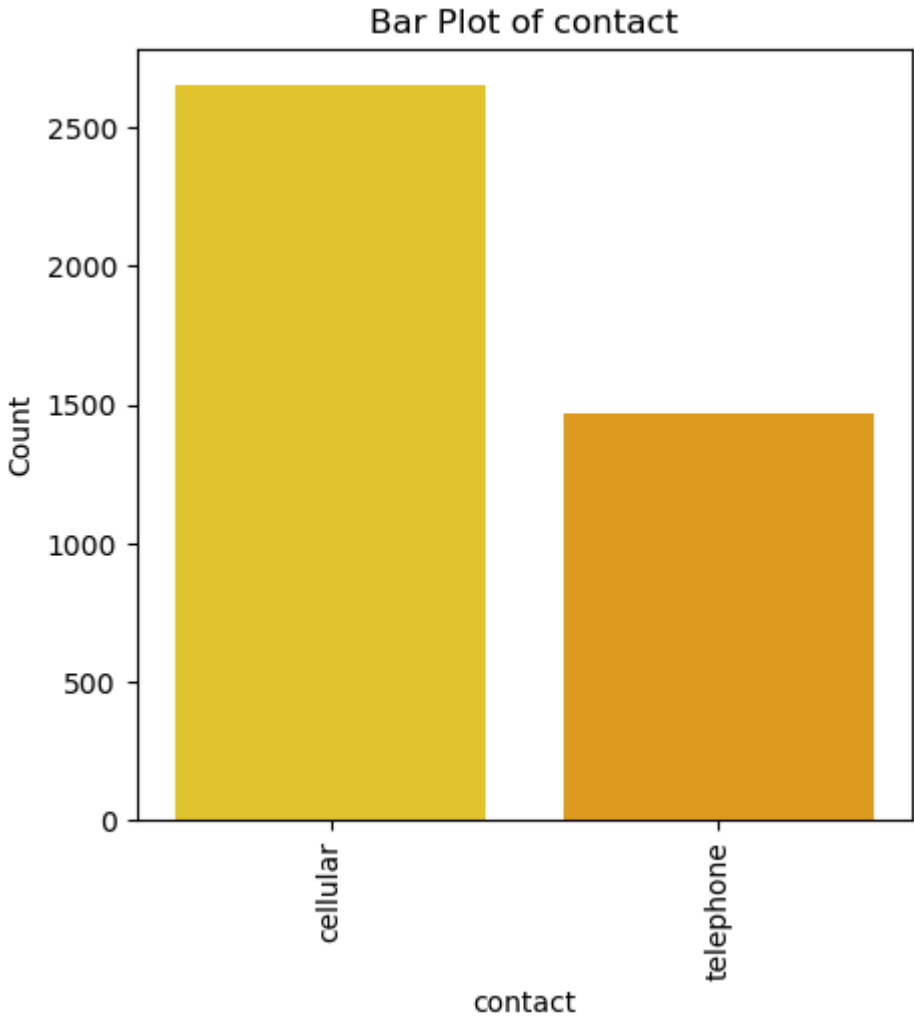


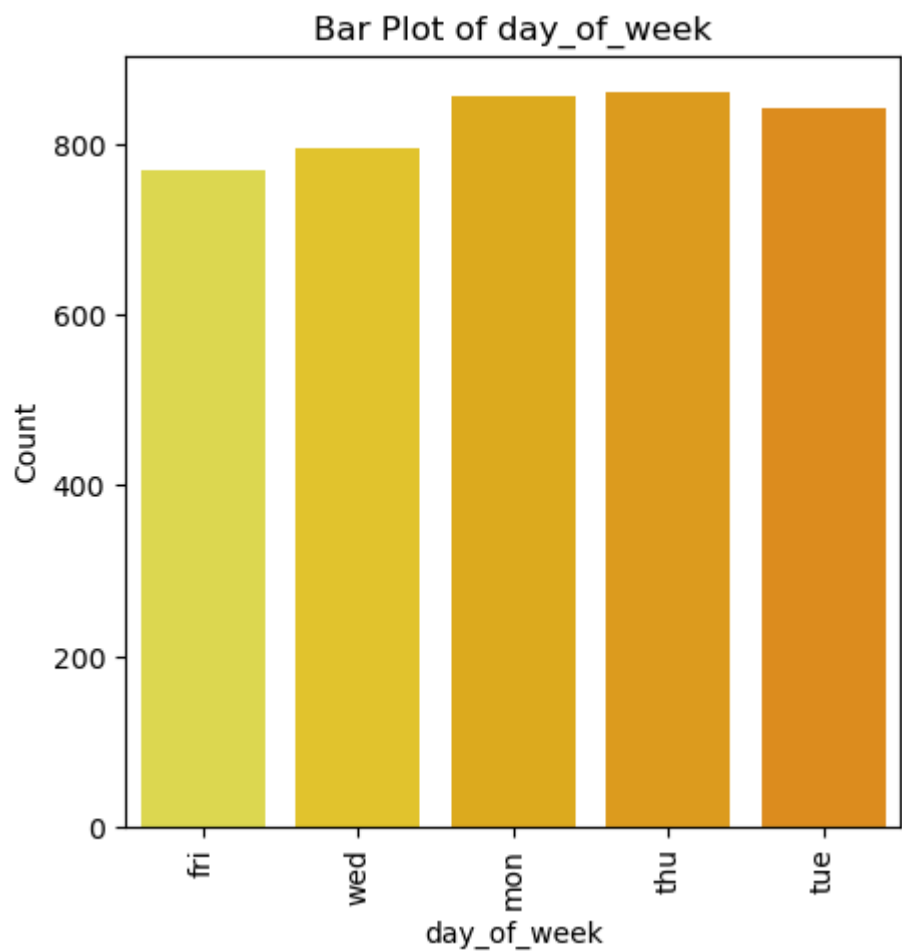


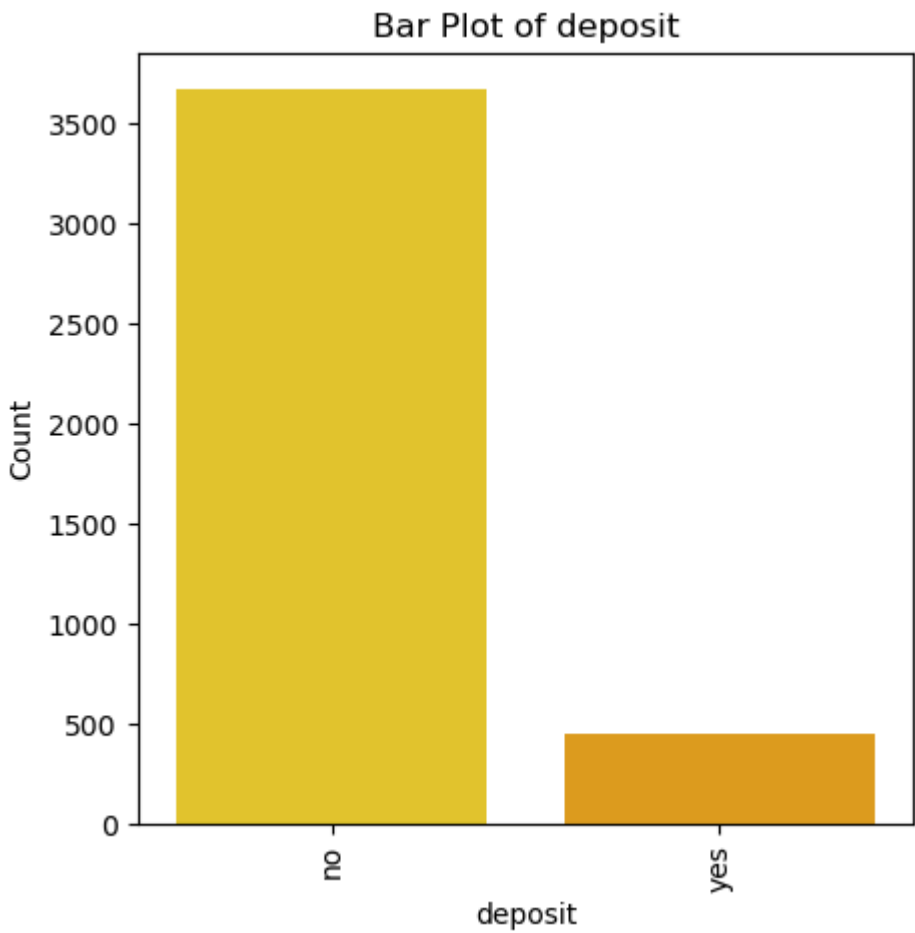
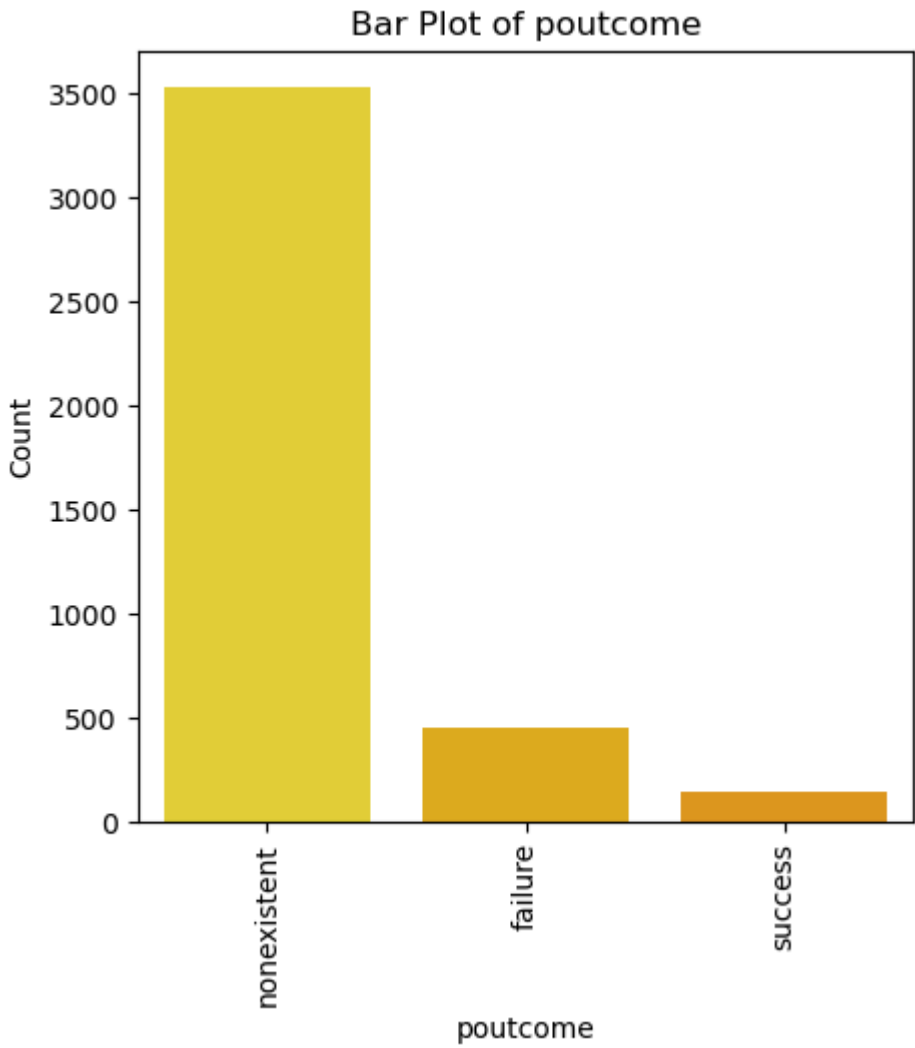




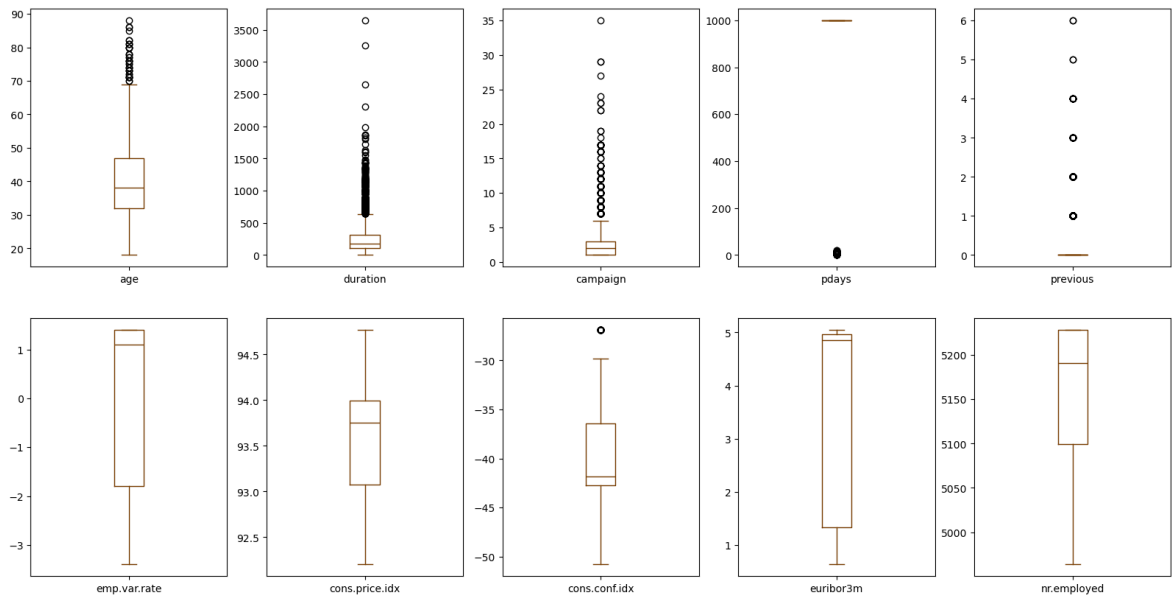






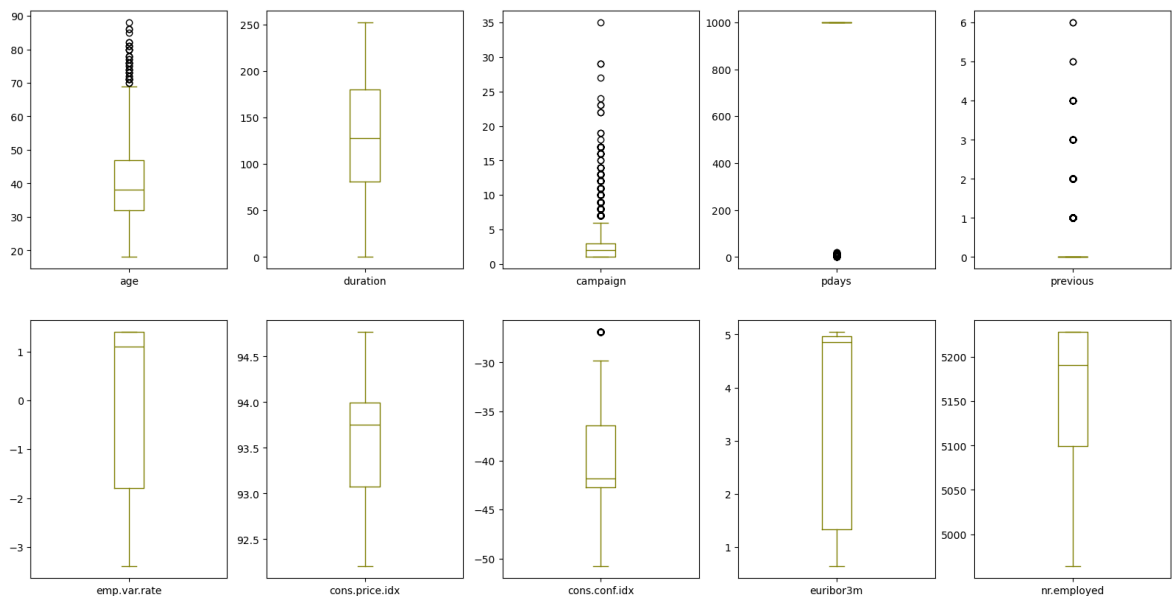


```
In [16]: df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00')
plt.show()
```



```
In [17]: 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 [18]: df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#808000')
plt.show()
```



```
In [19]: high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

```
In [20]: df1 = df.copy()
df1.columns
```

```
Out[20]: 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 [21]: df1.drop(high_corr_cols,inplace=True,axis=1) # axis=1 indicates columns
df1.columns
```

```
Out[21]: 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 [22]: df1.shape
```

```
Out[22]: (4119, 18)
```

```
In [23]: from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

```
Out[23]:
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_w
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	

4119 rows × 18 columns



```
In [24]: df_encoded['deposit'].value_counts()
```

```
Out[24]: deposit
0      3668
1       451
Name: count, dtype: int64
```

```
In [25]: 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'>
```

```
In [26]: from sklearn.model_selection import train_test_split

print(4119*0.25)
```

```
1029.75
```

```
In [27]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

```
In [28]: 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 [29]: from sklearn.tree import DecisionTreeClassifier

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

```
Out[29]: ▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

```
In [30]: mscore(dt)
```

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

```
In [31]: ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

```
[0 0 1 ... 0 0 0]
```

```
In [32]: eval_model(y_test,ypred_dt)
```

Accuracy\_Score 0.8990291262135922

Confusion Matrix

```
[[905 25]
 [ 79 21]]
```

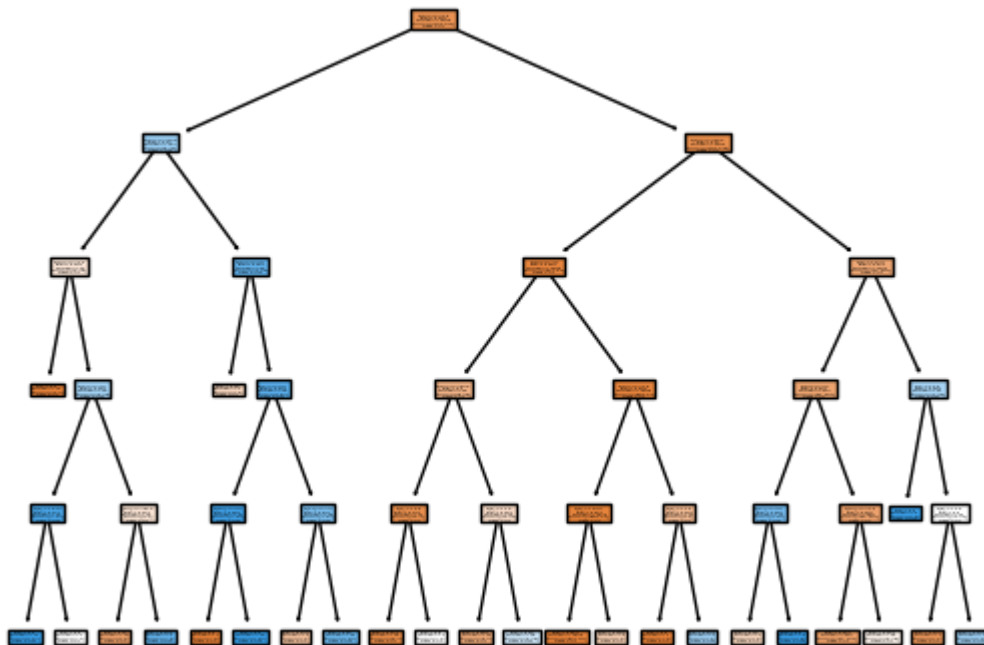
Classification Report

	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

```
In [35]: from sklearn.tree import plot_tree
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 [36]: plot_tree(dt, class_names=cn, filled=True)
plt.show()
```



```
In [37]: dt1 = DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=1)
dt1.fit(x_train, y_train)
```

Out[37]:

DecisionTreeClassifier

```
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=15)
```

In [38]: `mscore(dt1)`

Training Score 0.9080608611201036

Testing Score 0.9048543689320389

In [39]: `ypred_dt1 = dt1.predict(x_test)`  
`eval_model(y_test,ypred_dt1)`

Accuracy\_Score 0.9048543689320389

Confusion Matrix

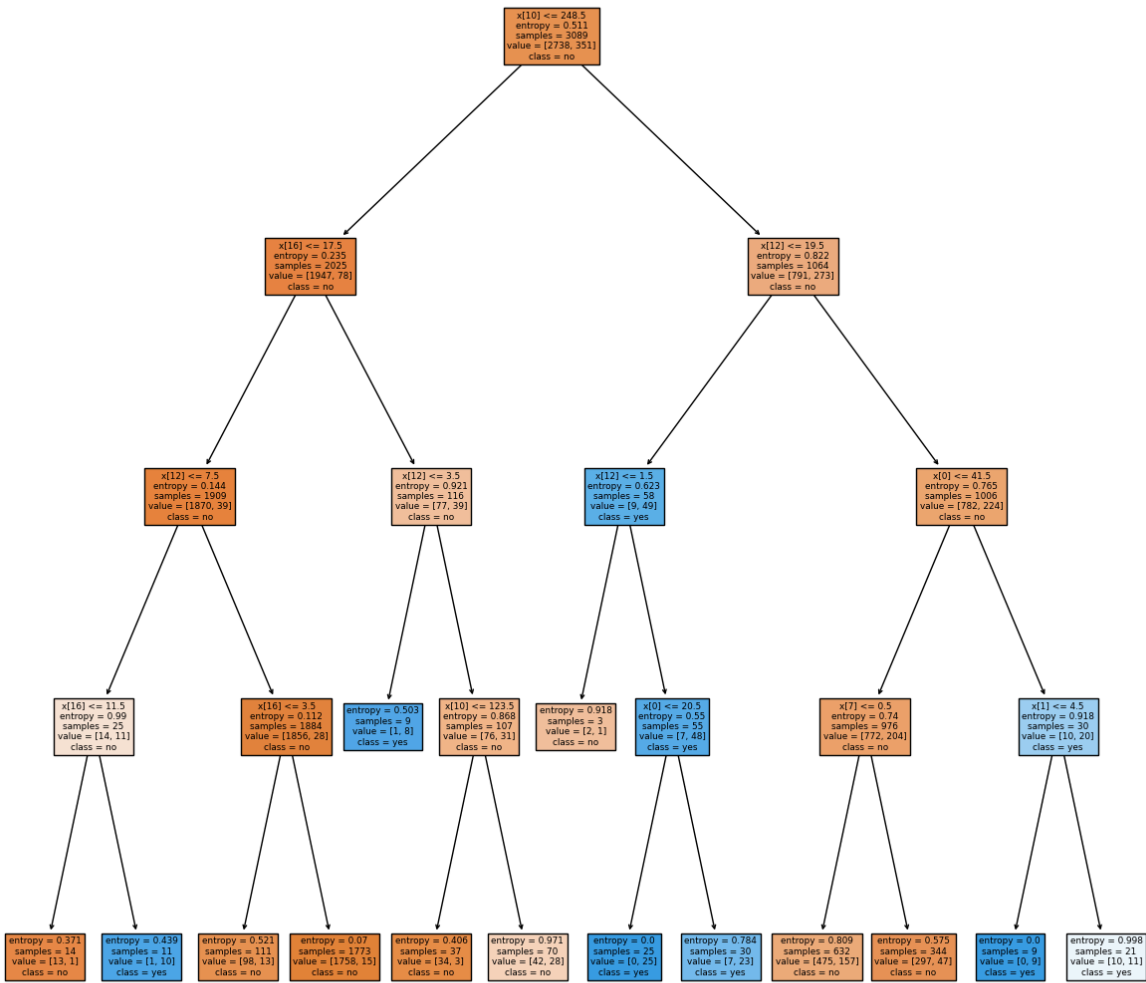
[[915 15]

[ 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

In [40]: `plt.figure(figsize=(15,15))`  
`plot_tree(dt1,class_names=cn,filled=True)`  
`plt.show()`



In [ ]: