```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline

# Load the dataset
df = pd.read_csv('bank.csv')

df.head()
```

```
   age         job  marital  education default  balance housing loan
contact  \
0   59      admin.  married  secondary      no     2343     yes   no
unknown
1   56      admin.  married  secondary      no       45      no   no
unknown
2   41  technician  married  secondary      no     1270     yes   no
unknown
3   55    services  married  secondary      no     2476     yes   no
unknown
4   54      admin.  married   tertiary      no      184      no   no
unknown

   day month  duration  campaign  pdays  previous poutcome deposit
0    5   may      1042         1     -1         0  unknown     yes
1    5   may      1467         1     -1         0  unknown     yes
2    5   may      1389         1     -1         0  unknown     yes
3    5   may       579         1     -1         0  unknown     yes
4    5   may       673         2     -1         0  unknown     yes
```

```
df.head()
```

```
   age         job  marital  education default  balance housing loan
contact  \
0   59      admin.  married  secondary      no     2343     yes   no
unknown
1   56      admin.  married  secondary      no       45      no   no
unknown
2   41  technician  married  secondary      no     1270     yes   no
unknown
3   55    services  married  secondary      no     2476     yes   no
unknown
4   54      admin.  married   tertiary      no      184      no   no
unknown
```

```
   day month  duration  campaign  pdays  previous poutcome deposit
0    5   may      1042         1     -1         0  unknown     yes
1    5   may      1467         1     -1         0  unknown     yes
2    5   may      1389         1     -1         0  unknown     yes
3    5   may       579         1     -1         0  unknown     yes
4    5   may       673         2     -1         0  unknown     yes
```

```
df.tail()
```

```
       age           job  marital  education default  balance housing
loan  \
11157   33  blue-collar   single    primary      no        1     yes
no
11158   39     services  married  secondary      no      733      no
no
11159   32   technician   single  secondary      no       29      no
no
11160   43   technician  married  secondary      no        0      no
yes
11161   34   technician  married  secondary      no        0      no
no

        contact  day month  duration  campaign  pdays  previous
poutcome  \
11157  cellular   20   apr       257         1     -1         0
unknown
11158   unknown   16   jun        83         4     -1         0
unknown
11159  cellular   19   aug       156         2     -1         0
unknown
11160  cellular    8   may         9         2    172         5
failure
11161  cellular    9   jul       628         1     -1         0
unknown

      deposit
11157      no
11158      no
11159      no
11160      no
11161      no
```

```
df.shape
```

```
(11162, 17)
```

```
df.columns
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance',
'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign',
```

```
'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

```
df.dtypes
```

```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
deposit      object
dtype: object
```

```
df.dtypes.value_counts()
```

```
object    10
int64      7
Name: count, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        11162 non-null  int64
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64
 12  campaign   11162 non-null  int64
```

```
 13  pdays      11162 non-null  int64
 14  previous   11162 non-null  int64
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

```
df.duplicated().sum()
```

```
0
```

```
df.isna().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
deposit      0
dtype: int64
```

```
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', 'poutcome', 'deposit'],
      dtype='object')
Index(['age', 'balance', 'day', 'duration', 'campaign', 'pdays',
'previous'], dtype='object')
```

```
df.describe()
```

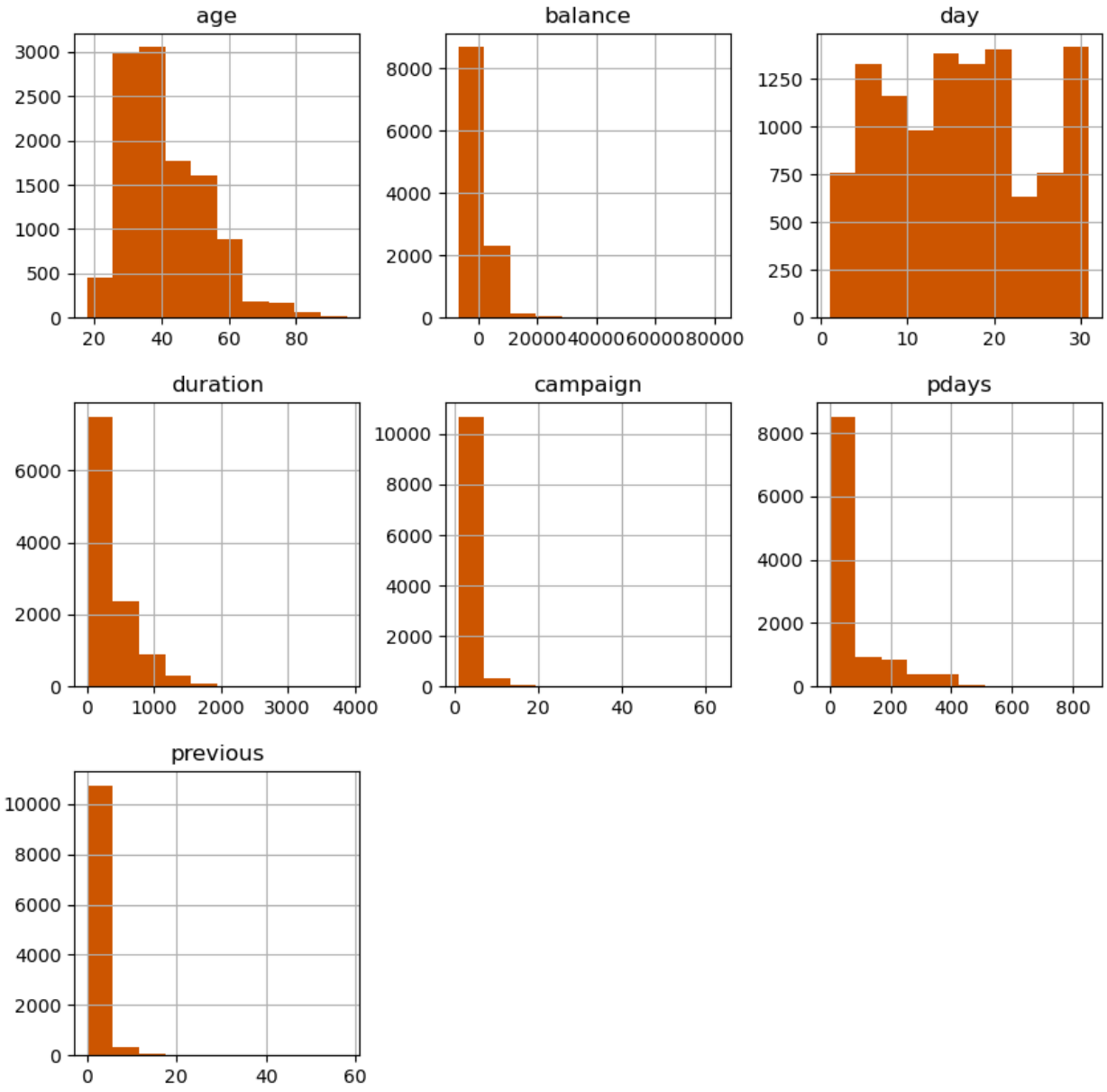
```
                age       balance           day      duration
campaign  \
count  11162.000000  11162.000000  11162.000000  11162.000000
11162.000000
```

```
mean      41.231948   1528.538524     15.658036    371.993818
2.508421
std       11.913369   3225.413326      8.420740    347.128386
2.722077
min       18.000000  -6847.000000      1.000000      2.000000
1.000000
25%       32.000000    122.000000      8.000000    138.000000
1.000000
50%       39.000000    550.000000     15.000000    255.000000
2.000000
75%       49.000000   1708.000000     22.000000    496.000000
3.000000
max       95.000000  81204.000000     31.000000   3881.000000
63.000000

              pdays      previous
count  11162.000000  11162.000000
mean      51.330407      0.832557
std      108.758282      2.292007
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       20.750000      1.000000
max      854.000000     58.000000
```

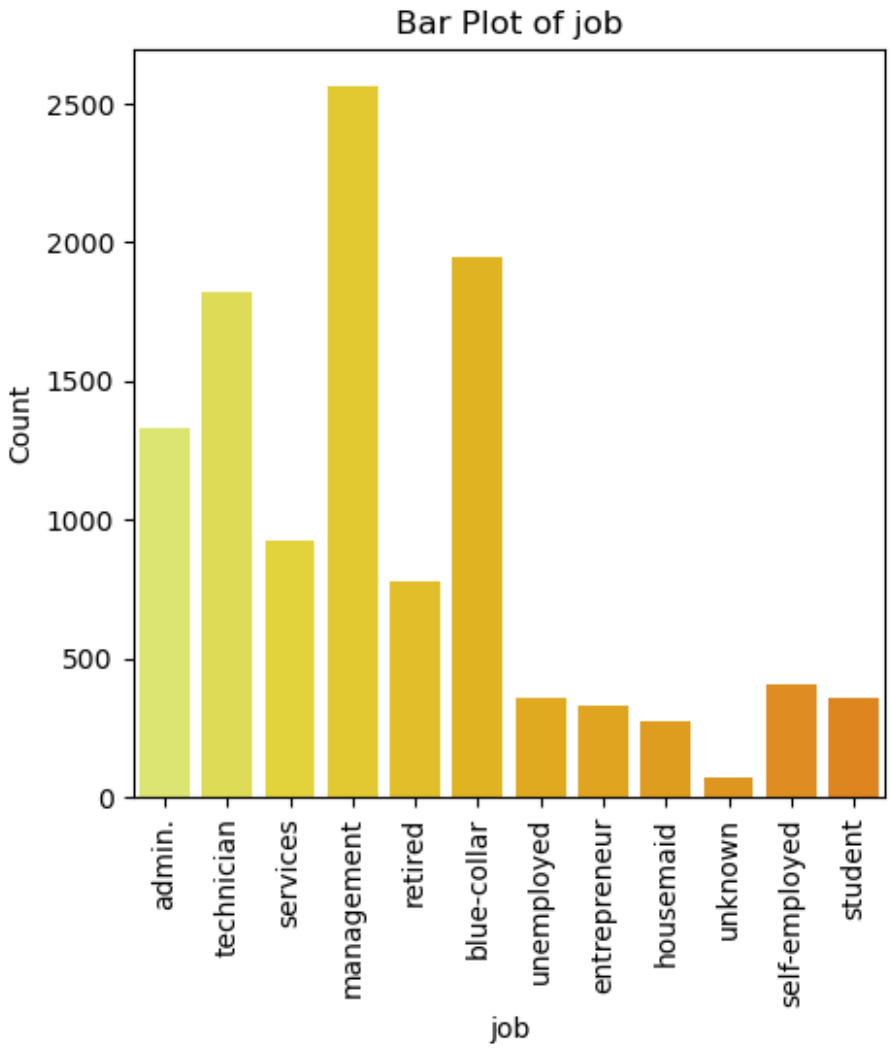
```
df.describe(include='object')
```

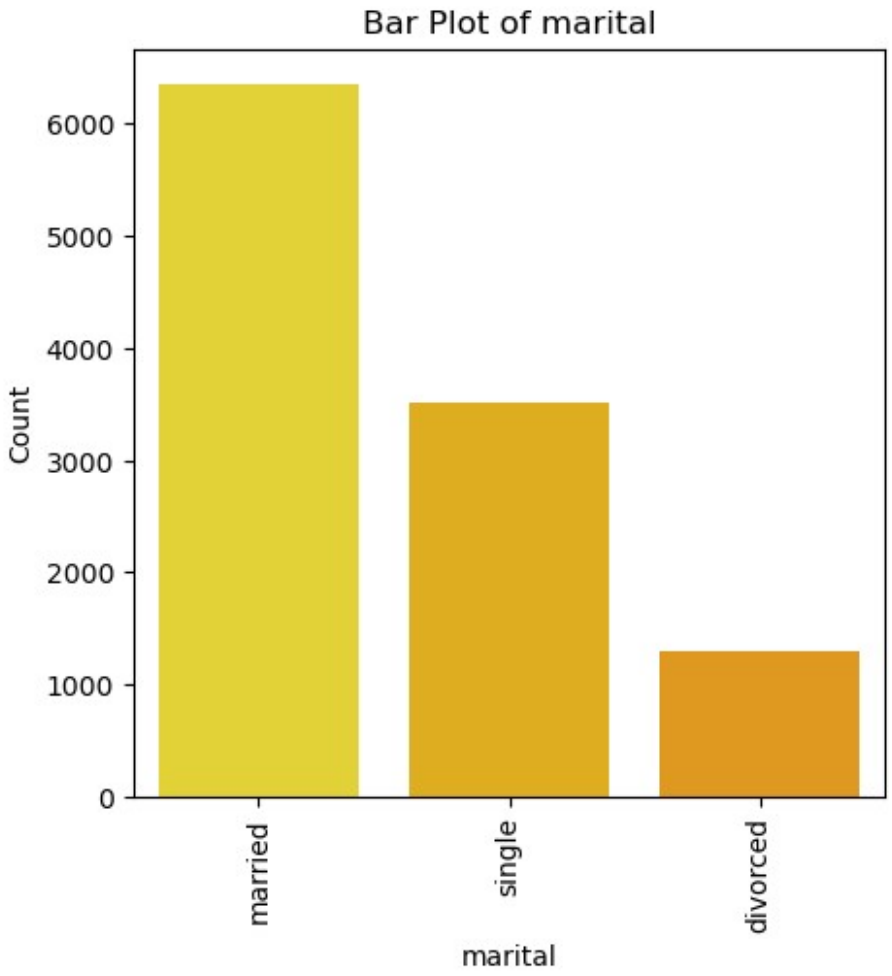
```
               job  marital  education default housing   loan
contact  \
count        11162    11162      11162   11162   11162  11162
11162
unique          12        3          4       2       2      2
3
top     management  married  secondary      no      no     no
cellular
freq          2566     6351       5476   10994    5881   9702
8042

        month poutcome deposit
count   11162    11162   11162
unique     12        4       2
top       may  unknown      no
freq     2824     8326    5873
```

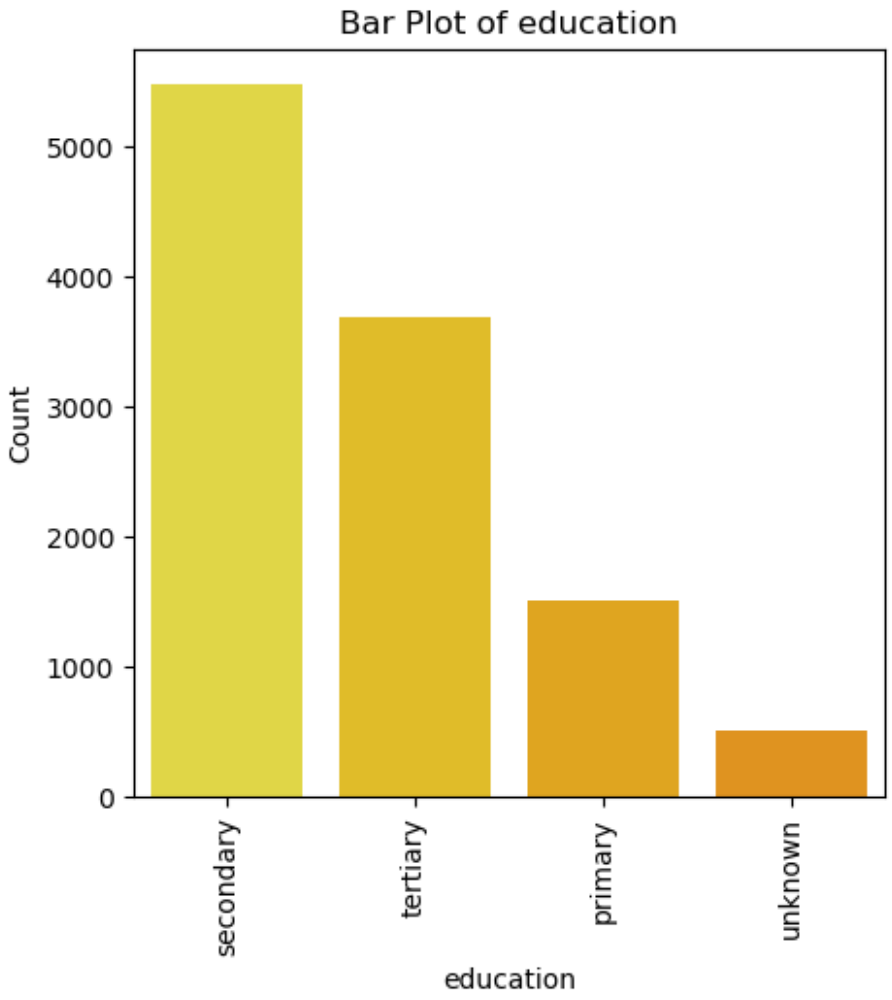
```
df.hist(figsize=(10,10),color='#cc5500')
plt.show()
```

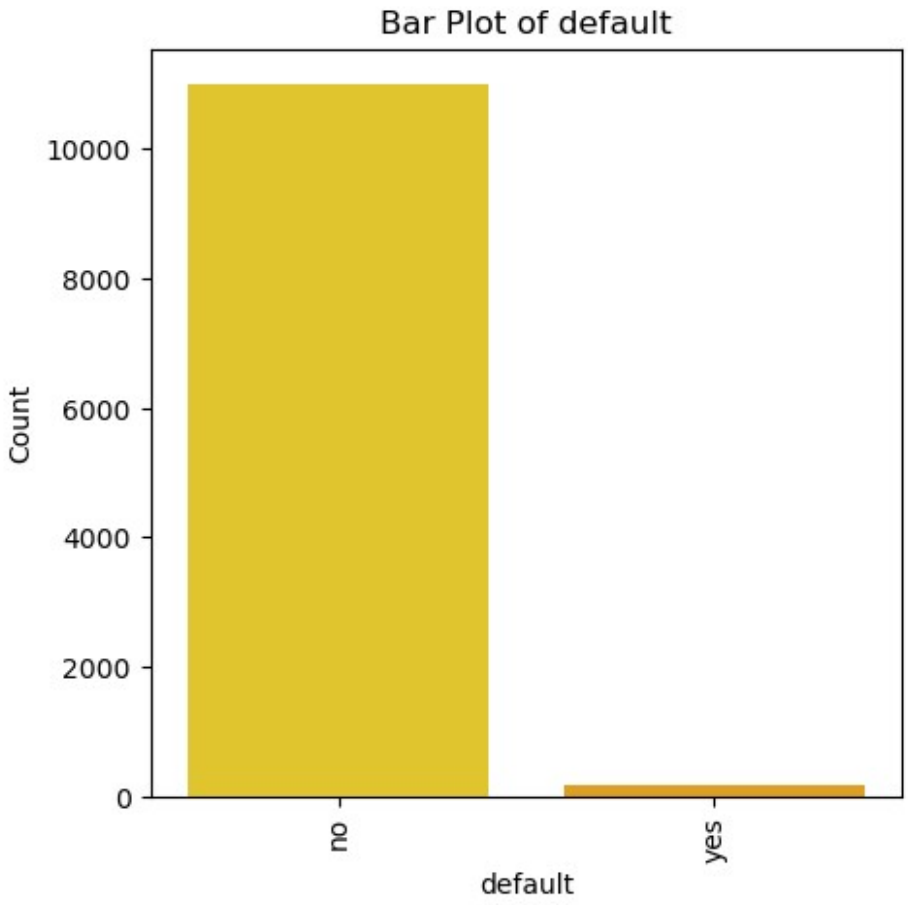


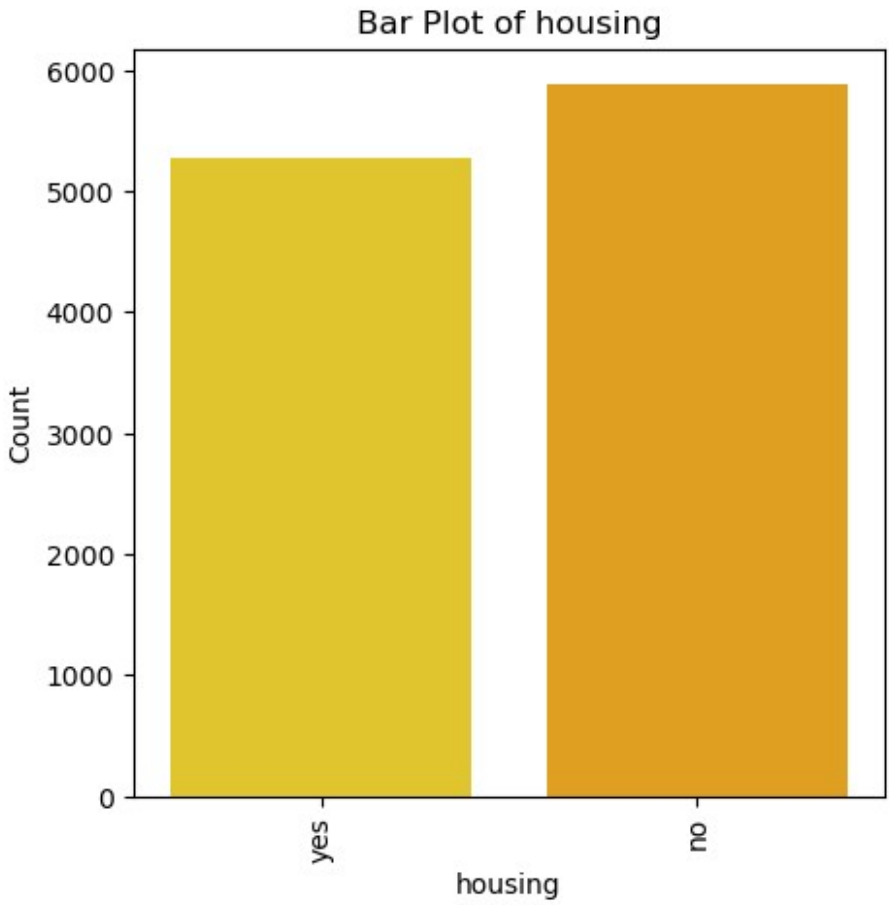

```
for feature in cat_cols:
    plt.figure(figsize=(5,5))  # Adjust the figure size as needed
    sns.countplot(x=feature, data=df, palette='Wistia')
    plt.title(f'Bar Plot of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=90)
    plt.show()
```

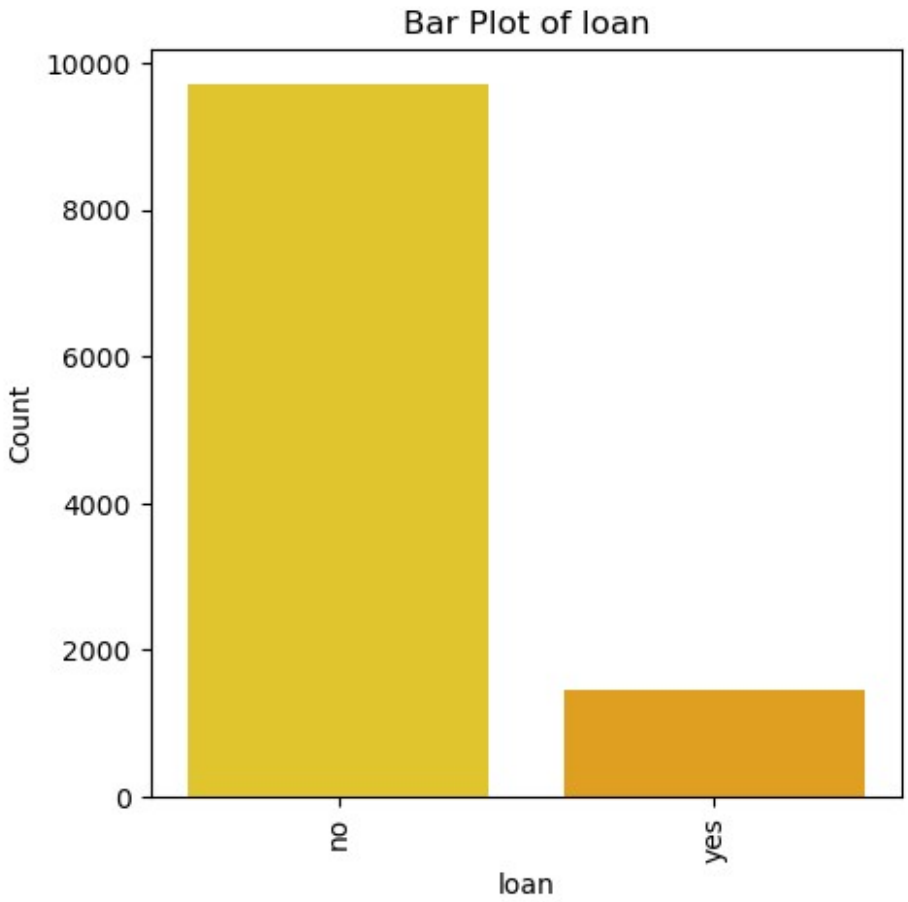

Bar Plot of job
Count
2500
2000
1500
1000
500
0
admin.
technician
services
management
retired
blue-collar
unemployed
entrepreneur
housemaid
unknown
self-employed
student
job

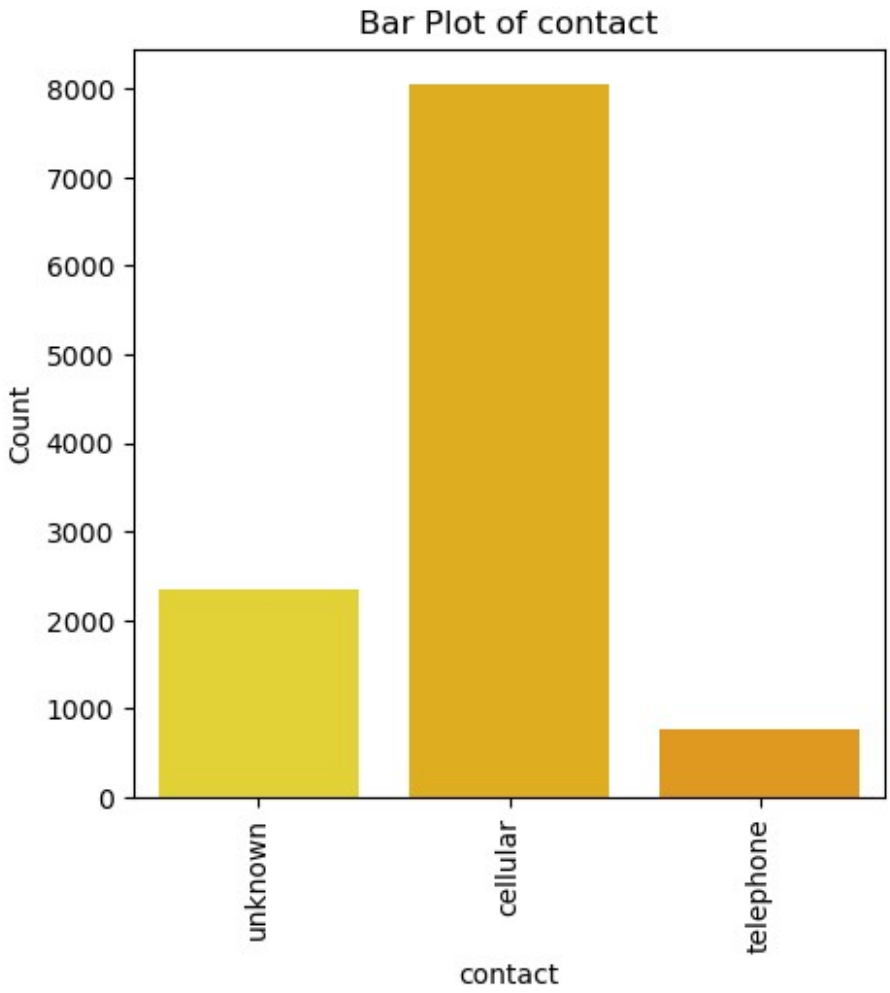

Bar Plot of marital
Count
6000
5000
4000
3000
2000
1000
0
married
single
divorced
marital

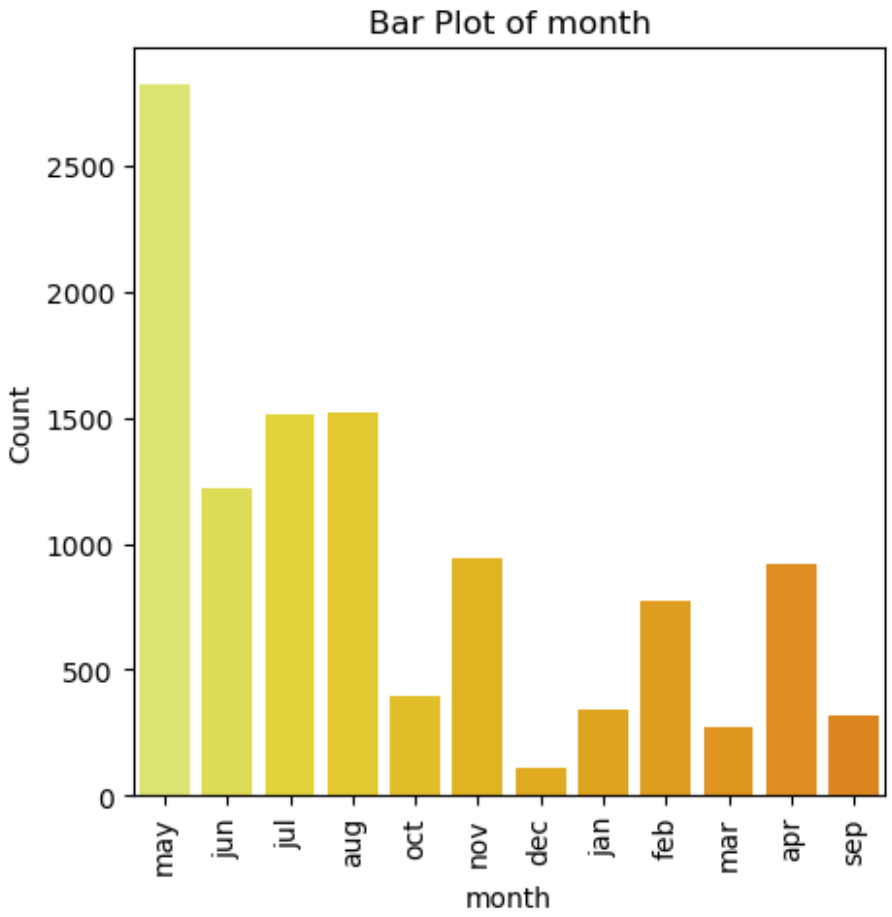

Bar Plot of education
Count
5000
4000
3000
2000
1000
0
secondary
tertiary
primary
unknown
education

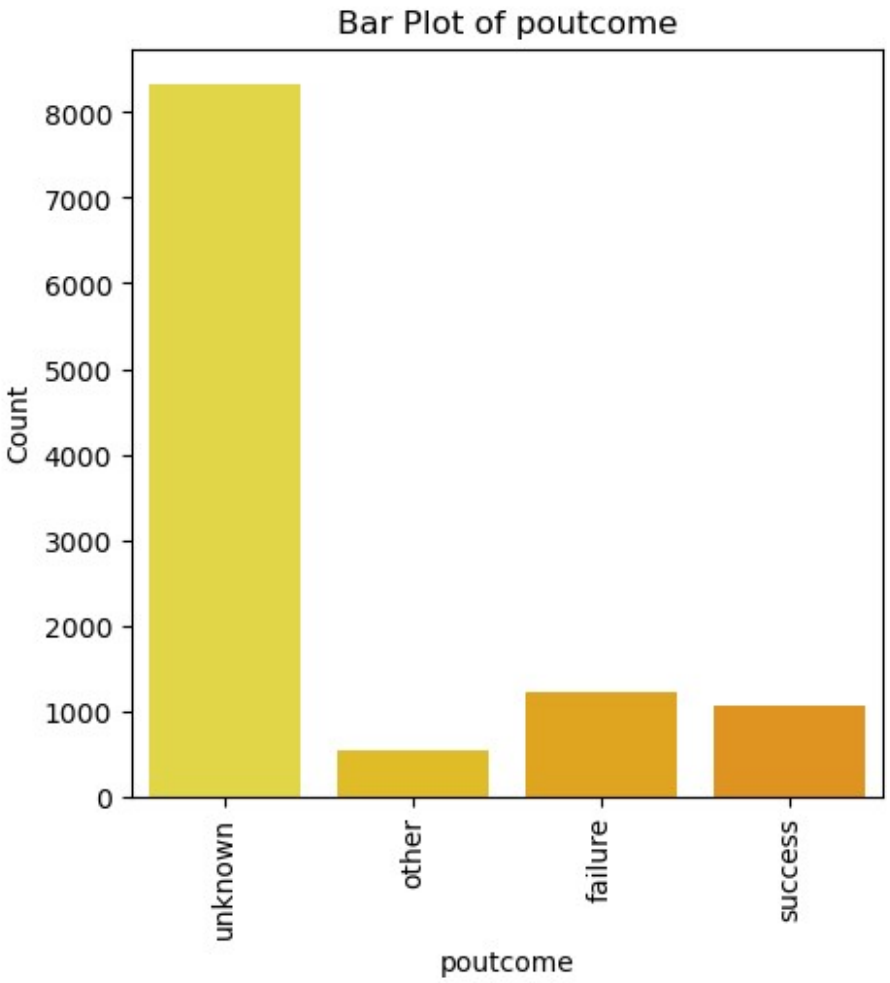

Bar Plot of default
Count
10000
8000
6000
4000
2000
0
no
yes
default

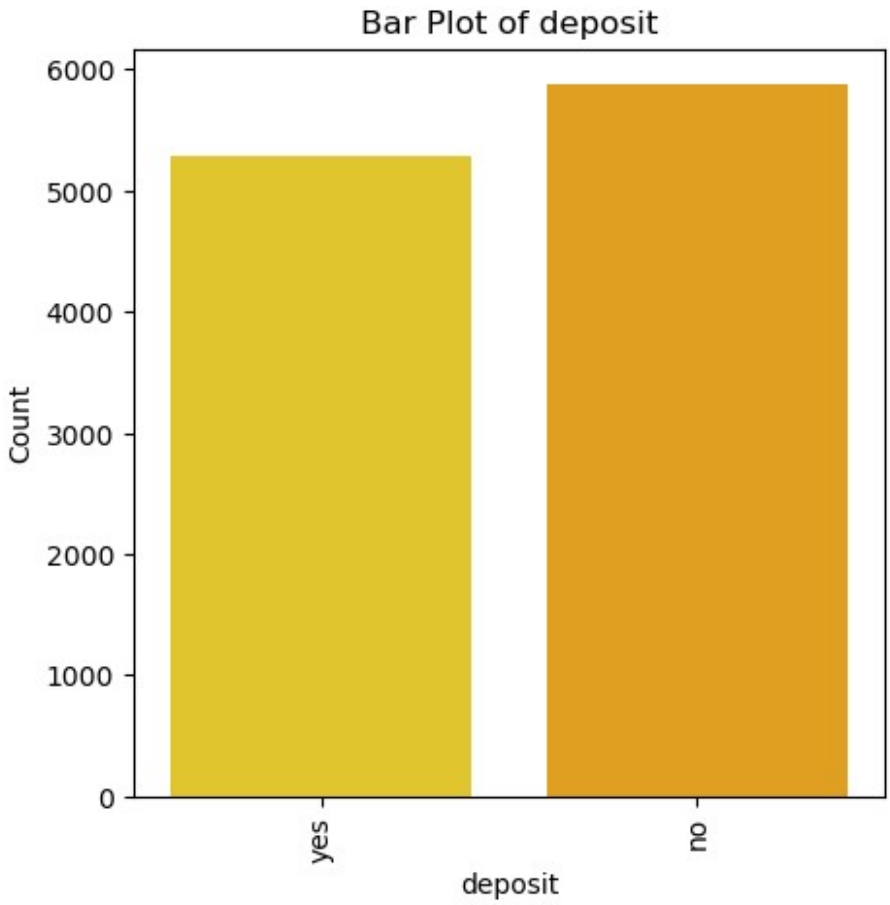

Bar Plot of housing
6000
5000
4000
3000
2000
1000
0
Count
yes
no
housing


Bar Plot of loan
10000
8000
6000
4000
2000
0
Count
no
yes
loan

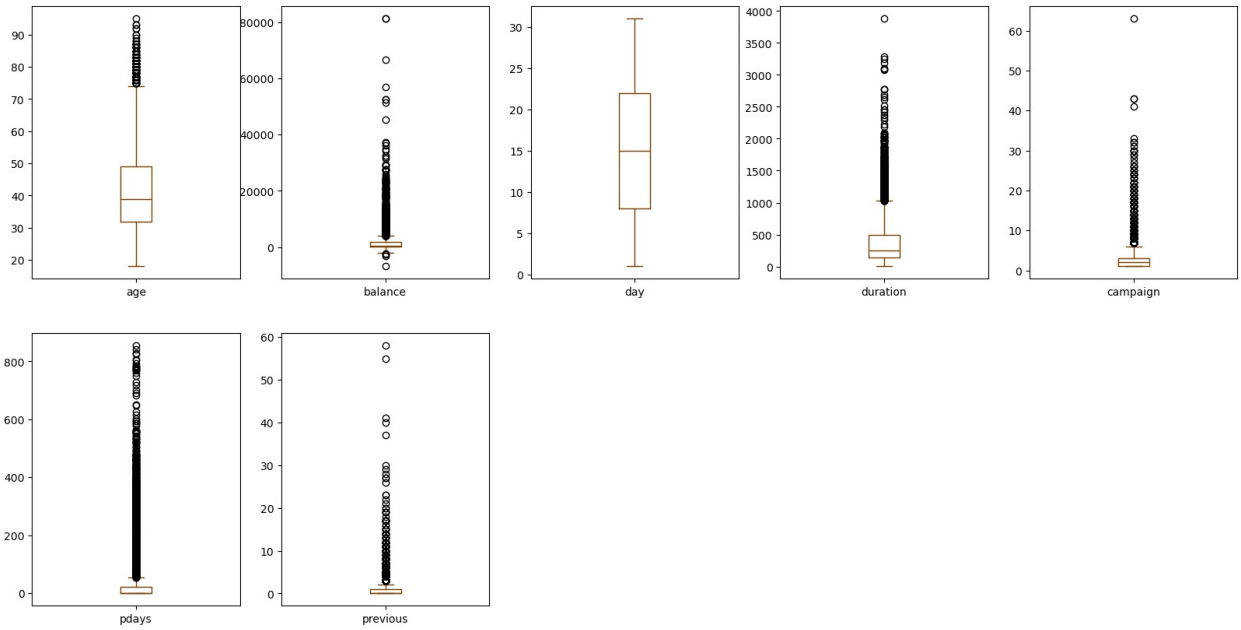

Bar Plot of contact
8000
7000
6000
5000
4000
3000
2000
1000
0
Count
unknown
cellular
telephone
contact


Bar Plot of month
Count
2500
2000
1500
1000
500
0
may
jun
jul
aug
oct
nov
dec
jan
feb
mar
apr
sep
month

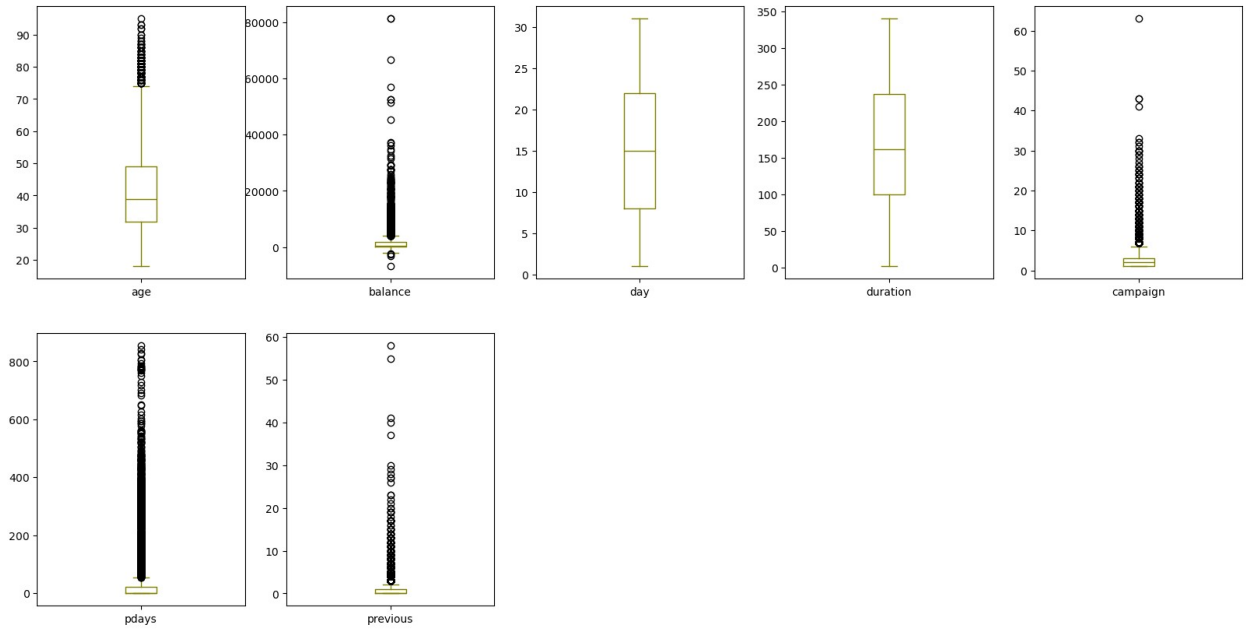

Bar Plot of poutcome
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
unknown
other
failure
success
poutcome



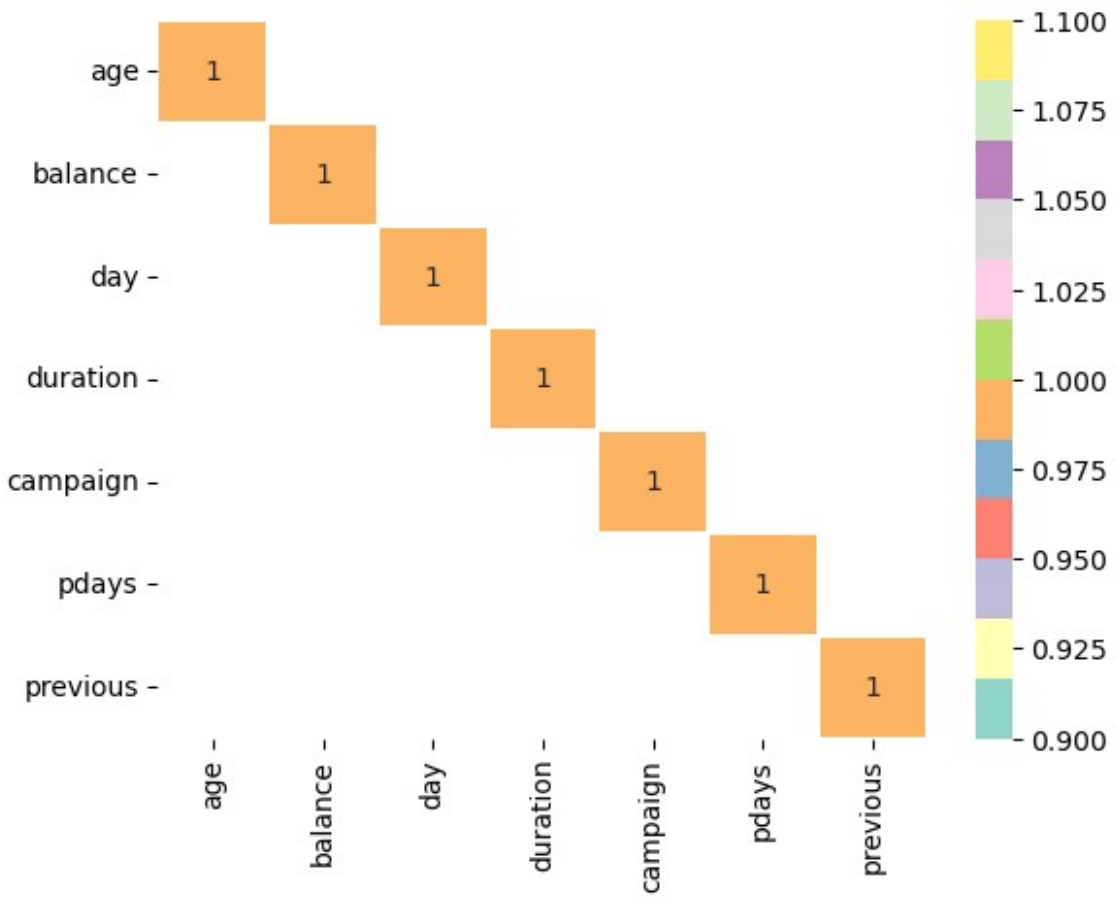

```
df.plot(kind='box', subplots=True,
layout=(2,5),figsize=(20,10),color='#7b3f00')
plt.show()
```




```
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)]

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




```
#Select only numeric columns
df_numeric = df.select_dtypes(include=['float64', 'int64'])

# Calculate correlation
corr = df_numeric.corr()

# Filter correlations
corr_filtered = corr[abs(corr) >= 0.90]

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




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

df1 = df.copy()
df1.columns

Index(['age', 'job', 'marital', 'education', 'default', 'balance',
'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign',
'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')

df1.shape

(11162, 17)

from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded

       age  job  marital  education  default  balance  housing  loan
contact  \
```

```
0       41    0        1           1        0     2288         1     0
2
1       38    0        1           1        0      469         0     0
2
2       23    9        1           1        0     1618         1     0
2
3       37    7        1           1        0     2356         1     0
2
4       36    0        1           2        0      608         0     0
2
...    ...  ...      ...         ...      ...      ...       ...   ...
...
11157   15    1        2           0        0      425         1     0
0
11158   21    7        1           1        0     1149         0     0
2
11159   14    9        2           1        0      453         0     0
0
11160   25    9        1           1        0      424         0     1
0
11161   16    9        1           1        0      424         0     0
0

       day  month  duration  campaign  pdays  previous  poutcome
deposit
0        4      8       339         0      0         0         3
1
1        4      8       339         0      0         0         3
1
2        4      8       339         0      0         0         3
1
3        4      8       339         0      0         0         3
1
4        4      8       339         1      0         0         3
1
...    ...    ...       ...       ...    ...       ...       ...
...
11157   19      0       255         0      0         0         3
0
11158   15      6        81         3      0         0         3
0
11159   18      1       154         1      0         0         3
0
11160    7      8         7         1    155         5         0
0
11161    8      5       339         0      0         0         3
0

[11162 rows x 17 columns]
```

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

```
deposit
0    5873
1    5289
Name: count, dtype: int64
```

```
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))
```

```
(11162, 16)
(11162,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

```
from sklearn.model_selection import train_test_split

print(4119*0.25)
```

```
1029.75
```

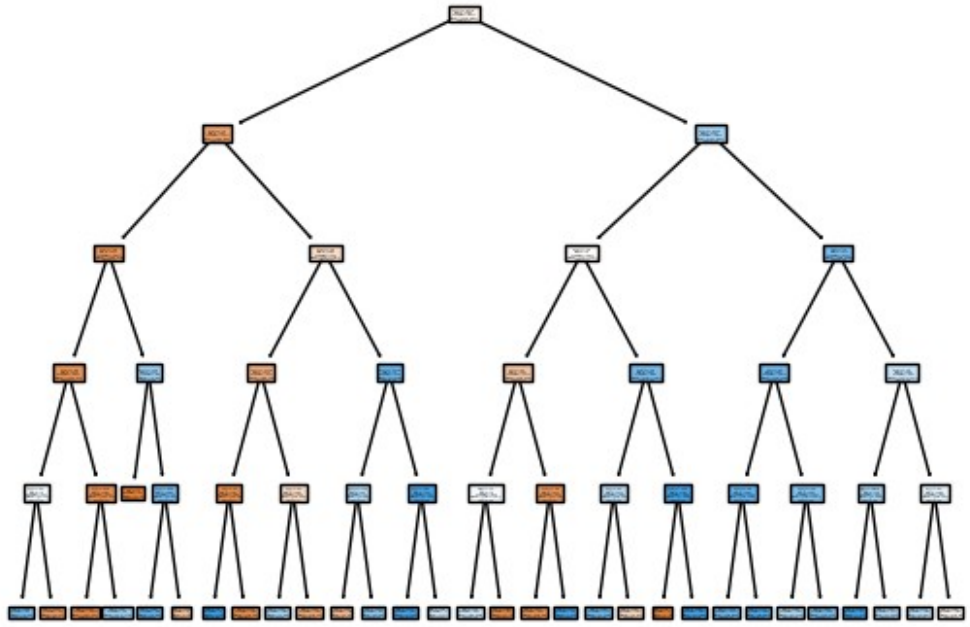
```
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.25,random_state=1)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(8371, 16)
(2791, 16)
(8371,)
(2791,)
```

```
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)
```

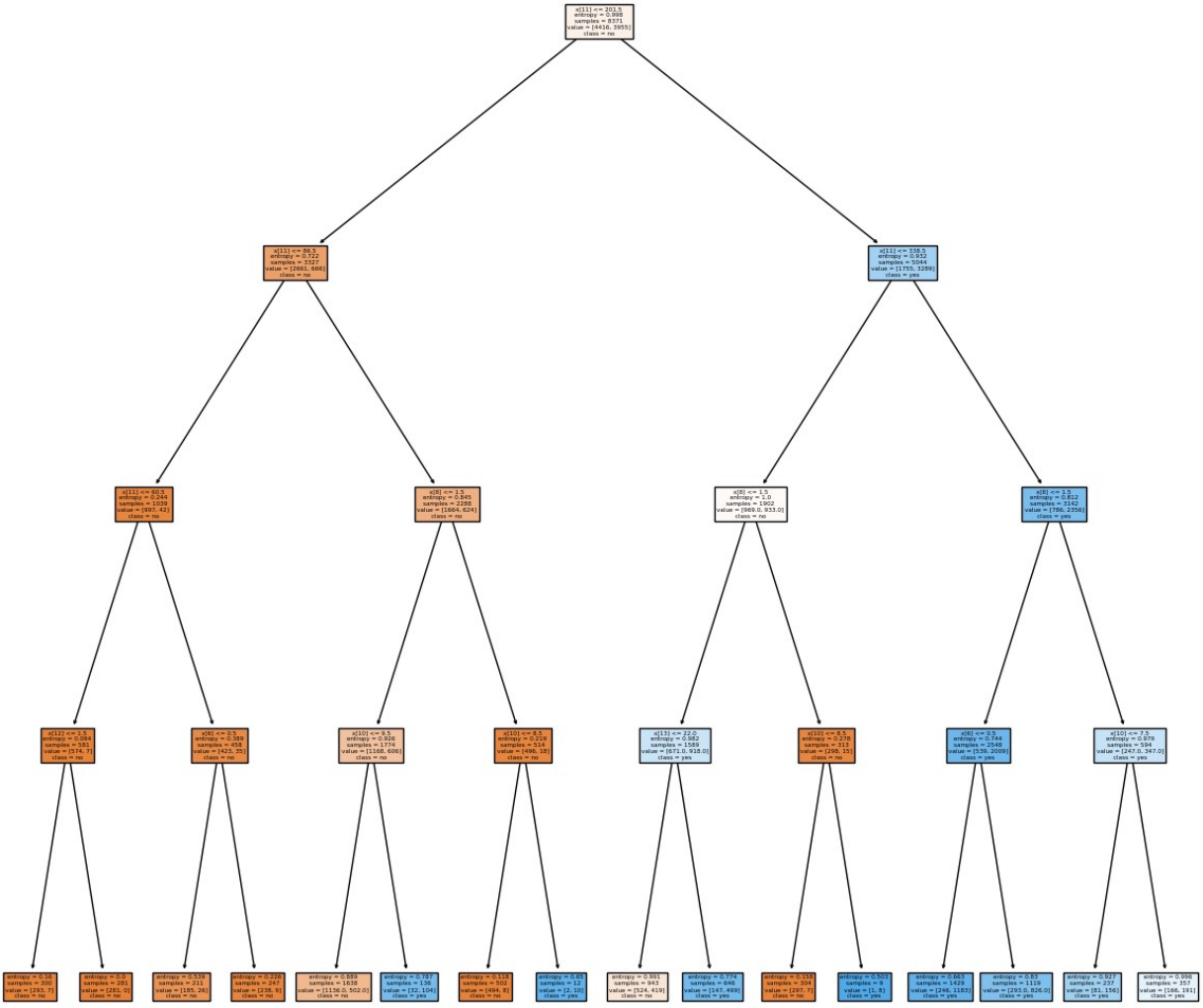
```
    print('Training Score',train_score)
    print('Testing Score',test_score)

from sklearn.tree import DecisionTreeClassifier

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

DecisionTreeClassifier(max_depth=5, min_samples_split=10)

mscore(dt)

Training Score 0.8058774339983276
Testing Score 0.7835901110713006

ypred_dt = dt.predict(x_test)
print(ypred_dt)

[0 0 1 ... 0 1 0]

eval_model(y_test,ypred_dt)

Accuracy_Score 0.7835901110713006
Confusion Matrix
 [[1047  410]
 [ 194 1140]]
Classification Report
              precision    recall  f1-score   support

           0       0.84      0.72      0.78      1457
           1       0.74      0.85      0.79      1334

    accuracy                           0.78      2791
   macro avg       0.79      0.79      0.78      2791
weighted avg       0.79      0.78      0.78      2791


from sklearn.tree import plot_tree

cn = ['no','yes']
fn = x_train.columns
print(fn)
print(cn)

Index(['age', 'job', 'marital', 'education', 'default', 'balance',
'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign',
'pdays',
       'previous', 'poutcome'],
```

```
      dtype='object')
['no', 'yes']

plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



```
dt1 =
DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_spl
it=15)
dt1.fit(x_train,y_train)

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

mscore(dt1)

Training Score 0.7675307609604587
Testing Score 0.7642422070942314

ypred_dt1 = dt1.predict(x_test)

eval_model(y_test,ypred_dt1)

Accuracy_Score 0.7642422070942314
Confusion Matrix
 [[1144  313]
 [ 345  989]]
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.77      0.79      0.78      1457
           1       0.76      0.74      0.75      1334

    accuracy                           0.76      2791
   macro avg       0.76      0.76      0.76      2791
weighted avg       0.76      0.76      0.76      2791


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


x[11] <= 201.5
entropy = 0.996
samples = 8371
value = [4416, 3955]
class = no
x[11] <= 86.5
entropy = 0.722
samples = 5227
value = [3661, 666]
class = no
x[11] <= 538.5
entropy = 0.652
samples = 3144
value = [755, 3289]
class = yes
x[11] <= 60.5
entropy = 0.34
samples = 1039
value = [997, 42]
class = no
x[8] <= 1.5
entropy = 0.845
samples = 4188
value = [2664, 624]
class = no
x[8] <= 1.5
entropy = 1.0
samples = 1902
value = [969.0, 933.0]
class = no
x[8] <= 1.5
entropy = 0.612
samples = 1142
value = [786, 2356]
class = yes
x[12] <= 5.5
entropy = 0.084
samples = 582
value = [574, 7]
class = no
x[6] <= 0.5
entropy = 0.369
samples = 458
value = [423, 51]
class = no
x[10] <= 9.5
entropy = 0.926
samples = 1774
value = [1169, 606]
class = no
x[10] <= 8.5
entropy = 0.314
samples = 514
value = [496, 18]
class = no
x[13] <= 22.0
entropy = 0.982
samples = 1589
value = [671.0, 918.0]
class = yes
x[10] <= 8.5
entropy = 0.235
samples = 313
value = [298, 15]
class = no
x[6] <= 0.5
entropy = 0.744
samples = 2548
value = [539, 2009]
class = yes
x[10] <= 7.5
entropy = 0.879
samples = 594
value = [247.0, 347.0]
class = yes
entropy = 0.16
samples = 390
value = [295, 7]
class = no
entropy = 0.0
samples = 281
value = [281, 0]
class = no
entropy = 0.539
samples = 211
value = [185, 26]
class = no
entropy = 0.226
samples = 247
value = [238, 9]
class = no
entropy = 0.889
samples = 1638
value = [1156.0, 502.0]
class = no
entropy = 0.787
samples = 136
value = [32, 104]
class = yes
entropy = 0.118
samples = 502
value = [494, 8]
class = no
entropy = 0.65
samples = 12
value = [2, 10]
class = yes
entropy = 0.991
samples = 943
value = [524, 419]
class = no
entropy = 0.774
samples = 646
value = [147, 490]
class = yes
entropy = 0.318
samples = 304
value = [297, 7]
class = no
entropy = 0.503
samples = 9
value = [1, 8]
class = yes
entropy = 0.663
samples = 1429
value = [246, 1183]
class = yes
entropy = 0.83
samples = 1119
value = [293.0, 826.0]
class = yes
entropy = 0.927
samples = 237
value = [81, 156]
class = yes
entropy = 0.996
samples = 357
value = [166, 191]
class = yes