TASK 3

Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioral data. Use a dataset such as the Bank Marketing dataset from the UCI Machine Learning Repository

In [106]:
```
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 [107]:
```
df = pd.read_csv("bank-additional.csv")
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

Out[107]:

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

5 rows × 21 columns

In [108]:
```
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 [109]:
```
df.tail()
```

Out[109]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_w
4114	30	admin.	married	basic.6y	no	yes	yes	cellular	jul	
4115	39	admin.	married	high.school	no	yes	no	telephone	jul	
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 [110]:
```
df.shape
```

Out[110]: (4119, 21)

In [111]:
```
df.columns
```

Out[111]:
```
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 [112]:
```
df.dtypes
```

Out[112]:
```
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 [113]:
```
df.dtypes.value_counts()
```

Out[113]:
```
object     11
int64       5
float64     5
dtype: int64
```

In [114]:
```
df.duplicated().sum()
```

Out[114]:
```
0
```

In [115]:
```
df.isna().sum()
```

Out[115]:
```
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 [116]:
```
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', 'contac
t',
       '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 [117]:
```
df.describe()
```

Out[117]:

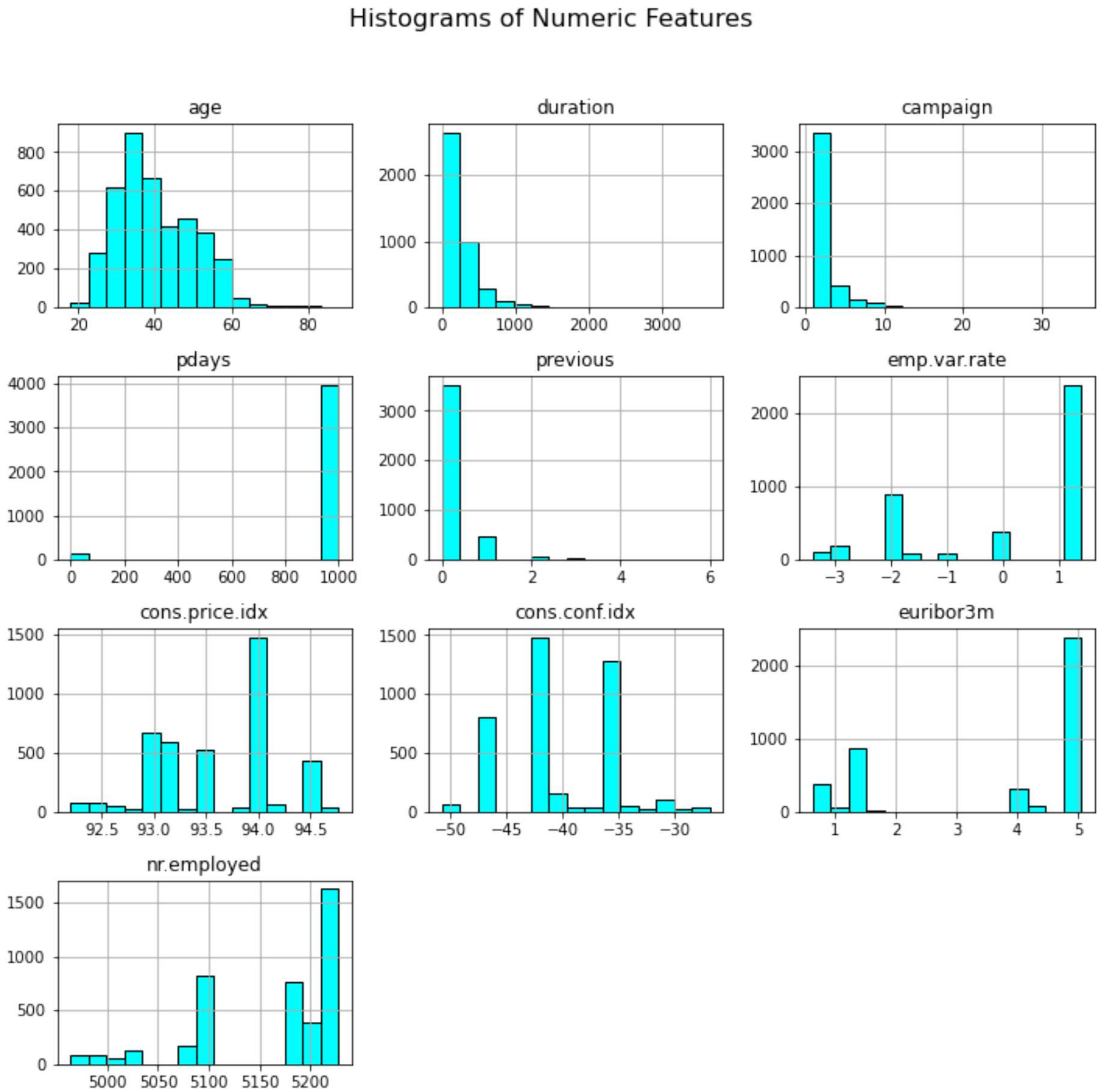
	age	duration	campaign	pdays	previous	emp.var.rate	cons.price
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.00
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.57
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.57
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.20
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.07
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.74
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.99
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.76

In [118]:
```
df.describe(include='object')
```

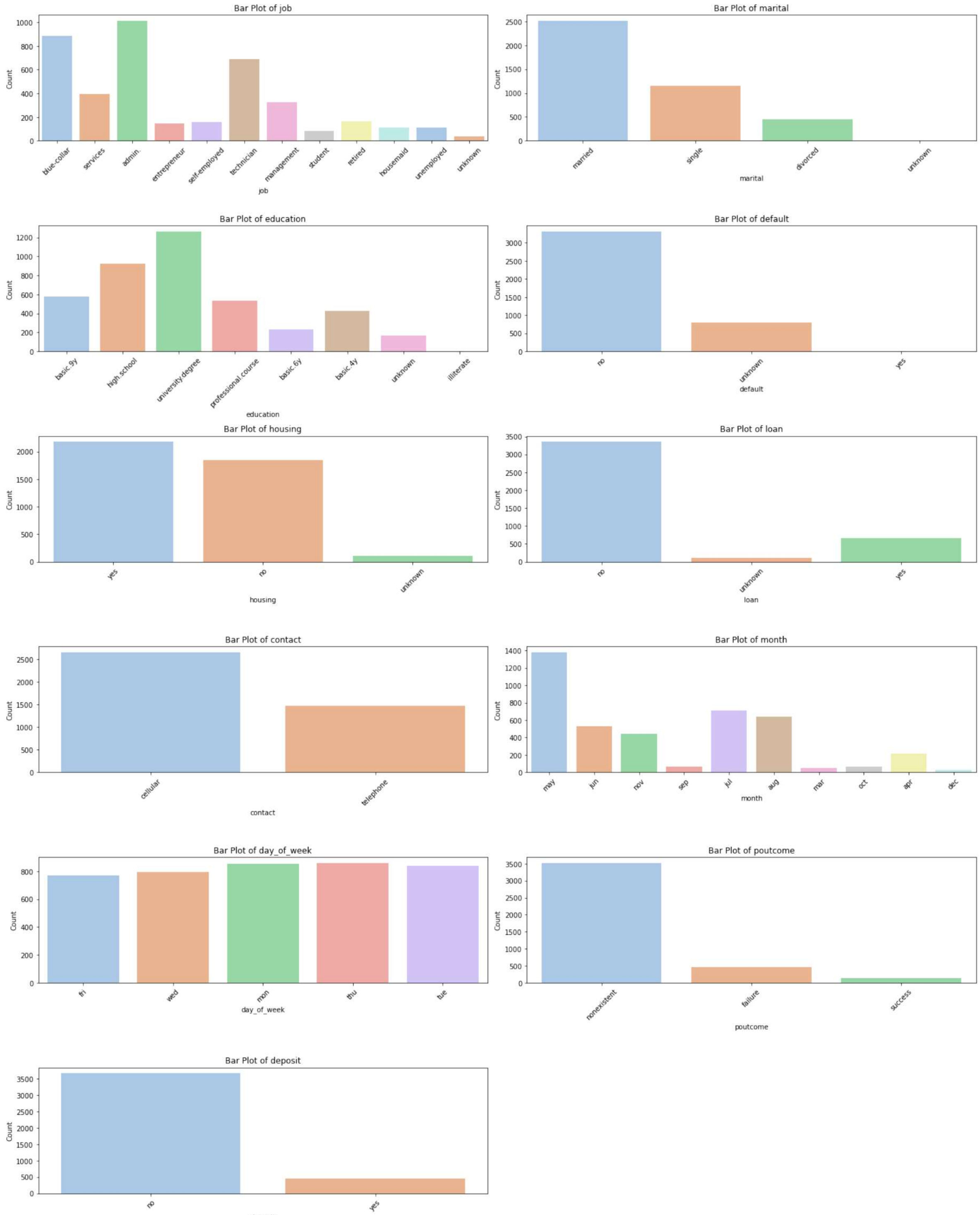
Out[118]:

	job	marital	education	default	housing	loan	contact	month	day_of_week
count	4119	4119	4119	4119	4119	4119	4119	4119	4119
unique	12	4	8	3	3	3	2	10	5
top	admin.	married	university.degree	no	yes	no	cellular	may	thu
freq	1012	2509	1264	3315	2175	3349	2652	1378	860

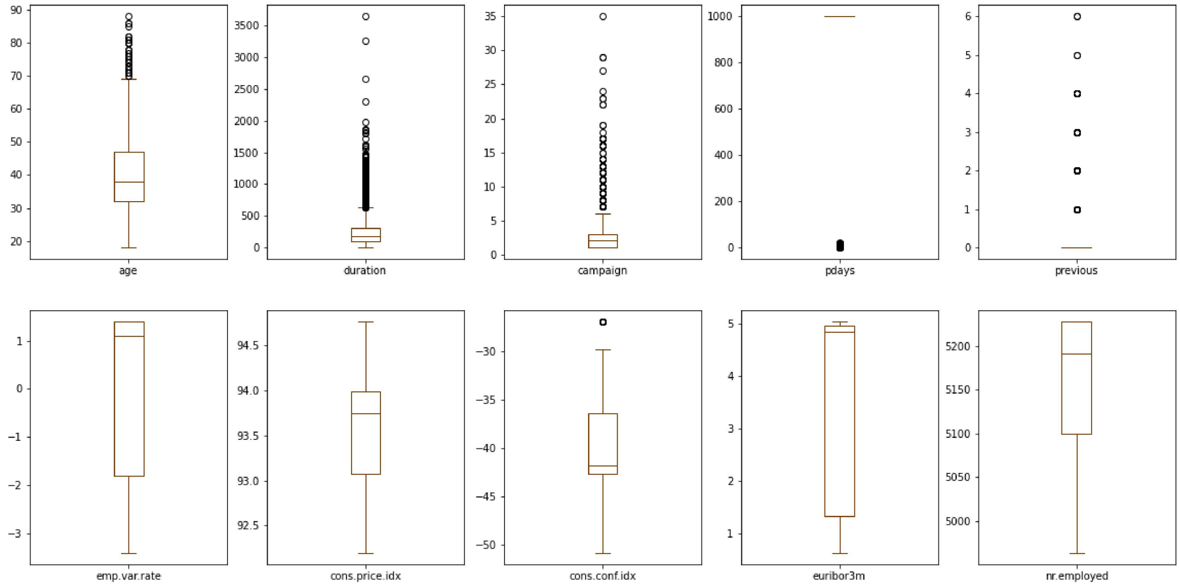
In [119]:
```
import matplotlib.pyplot as plt
df.hist(figsize=(10, 10), color='#00FFFF', edgecolor='black', bins=15)
plt.suptitle('Histograms of Numeric Features', fontsize=16)
plt.xlabel('Value', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```



In [120]:
```
num_plots = len(cat_cols)
num_rows = (num_plots + 1) // 2
num_cols = 2

plt.figure(figsize=(20, 25))

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

plt.tight_layout()
plt.show()
```

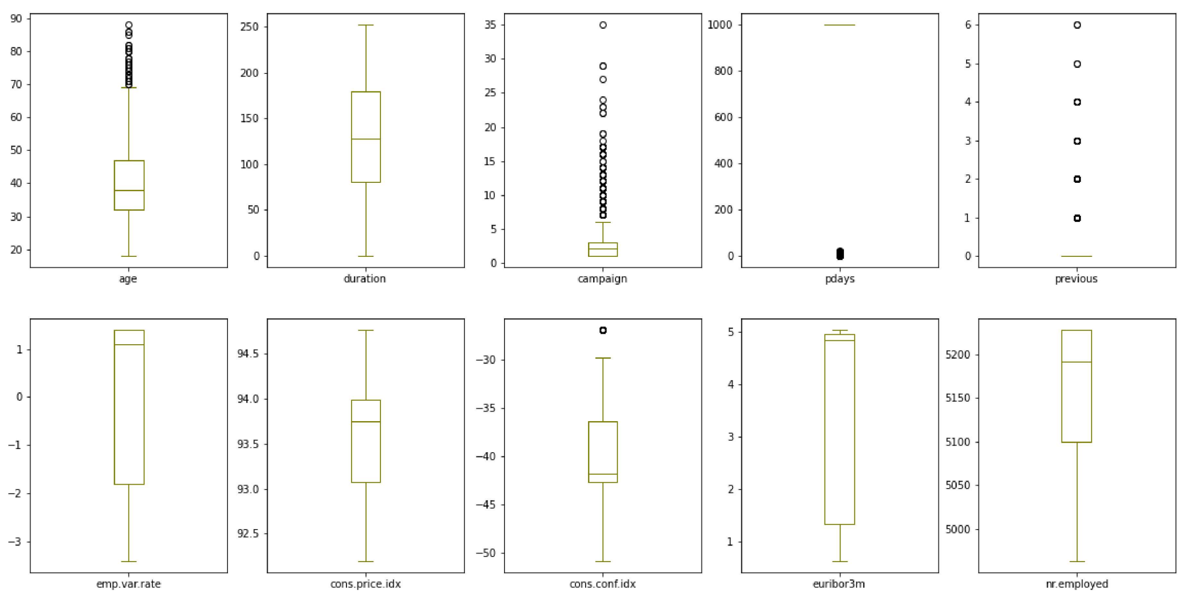

Bar Plot of job
Count
job
blue-collar
services
admin
entrepreneur
self-employed
technician
management
student
retired
housemaid
unemployed
unknown
Bar Plot of marital
marital
married
single
divorced
unknown
Bar Plot of education
education
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate
Bar Plot of default
default
no
unknown
yes
Bar Plot of housing
housing
yes
no
unknown
Bar Plot of loan
loan
no
unknown
yes
Bar Plot of contact
contact
cellular
telephone
Bar Plot of month
month
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
Bar Plot of day_of_week
day_of_week
fri
wed
mon
thu
tue
Bar Plot of poutcome
poutcome
nonexistent
failure
success
Bar Plot of deposit
deposit
no
yes

In [121]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00
plt.show()
```



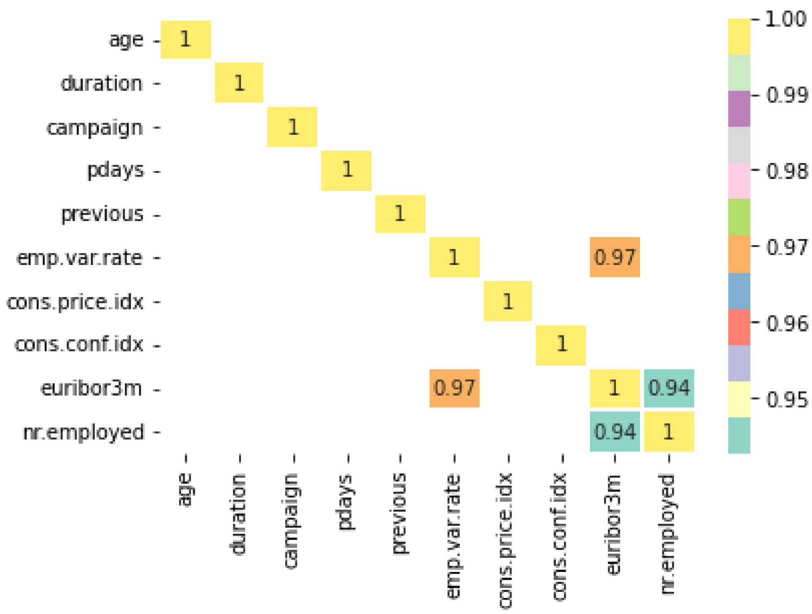
In [122]:
```
column = df[['age','campaign','duration']]
q1 = np.percentile(column, 25)
q3 = np.percentile(column, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
df[['age','campaign','duration']] = column[(column > lower_bound) & (column <
```

In [123]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#808000
plt.show()
```



In [124]:
```
numeric_df = df.drop(columns=cat_cols)
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  \
age            1.000000  0.014048 -0.014169 -0.043425  0.050931
duration       0.014048  1.000000 -0.218111 -0.093694  0.094206
campaign      -0.014169 -0.218111  1.000000  0.058742 -0.091490
pdays         -0.043425 -0.093694  0.058742  1.000000 -0.587941
previous       0.050931  0.094206 -0.091490 -0.587941  1.000000
emp.var.rate  -0.019192 -0.063870  0.176079  0.270684 -0.415238
cons.price.idx -0.000482 -0.013338  0.145021  0.058472 -0.164922
cons.conf.idx  0.098135  0.045889  0.007882 -0.092090 -0.051420
euribor3m     -0.015033 -0.067815  0.159435  0.301478 -0.458851
nr.employed   -0.041936 -0.097339  0.161037  0.381983 -0.514853

               emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
age               -0.019192       -0.000482       0.098135  -0.015033
duration          -0.063870       -0.013338       0.045889  -0.067815
campaign           0.176079        0.145021       0.007882   0.159435
pdays              0.270684        0.058472      -0.092090   0.301478
previous          -0.415238       -0.164922      -0.051420  -0.458851
emp.var.rate       1.000000        0.755155       0.195022   0.970308
cons.price.idx     0.755155        1.000000       0.045835   0.657159
cons.conf.idx      0.195022        0.045835       1.000000   0.276595
euribor3m          0.970308        0.657159       0.276595   1.000000
nr.employed        0.897173        0.472560       0.107054   0.942589

               nr.employed
age              -0.041936
duration         -0.097339
campaign          0.161037
pdays             0.381983
previous         -0.514853
emp.var.rate      0.897173
cons.price.idx    0.472560
cons.conf.idx     0.107054
euribor3m         0.942589
nr.employed       1.000000
```



In [125]:
```
high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

In [126]:
```
df1 = df.copy()
df1.columns
```

Out[126]:
```
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 [127]:
```
df1.drop(high_corr_cols,inplace=True,axis=1)  # axis=1 indicates columns
df1.columns
```

Out[127]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'cons.price.idx', 'cons.conf.idx', 'deposi
t'],
      dtype='object')
```

In [128]:
```
df1.shape
```

Out[128]:
```
(4119, 18)
```

In [129]:
```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

Out[129]:

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

4119 rows × 18 columns

In [130]:
```
df_encoded['deposit'].value_counts()
```

Out[130]:
```
0    3668
1     451
Name: deposit, dtype: int64
```

In [131]:
```
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 [132]:
```
from sklearn.model_selection import train_test_split
```

In [133]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_sta
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [134]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_sc

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 [135]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(criterion='gini',max_depth=5,min_samples_split=10)
dt.fit(x_train,y_train)
```

Out[135]:
```
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

In [136]:
```
mscore(dt)
```

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

In [137]:
```
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

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

In [138]:
```
eval_model(y_test,ypred_dt)
```

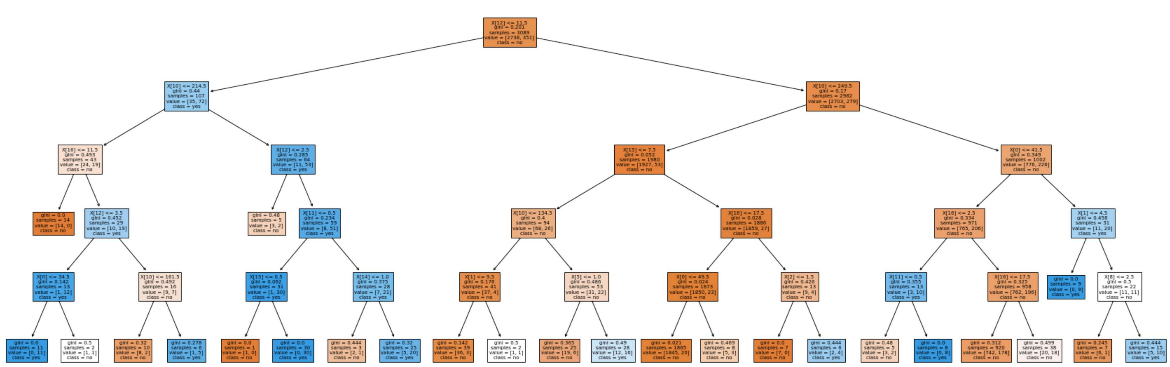
```
Accuracy_Score 0.8990291262135922
Confusion Matrix
 [[905  25]
 [ 79  21]]
Classification Report
               precision    recall  f1-score   support

           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 [139]:
```
from sklearn.tree import plot_tree
```

In [140]:
```
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 [141]:
```
plt.figure(figsize=(30,10))
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



In [142]:
```
dt1=DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=1
dt1.fit(x_train,y_train)
```

Out[142]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=1
5)
```

In [143]:
```
mscore(dt1)
```

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

In [144]:
```
ypred_dt1 = dt1.predict(x_test)
```

In [145]:
```
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
```