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 [4]:
```
df = pd.read_csv(r'C:\Users\Lenovo\Desktop\SURAJ TASK DATASET\bank-full.csv',del
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact	da
0	58	management	married	tertiary	no	2143	yes	no	unknown	
1	44	technician	single	secondary	no	29	yes	no	unknown	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	
4	33	unknown	single	unknown	no	1	no	no	unknown	

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

Out[5]:

	age	job	marital	education	default	balance	housing	loan	contact	da
0	58	management	married	tertiary	no	2143	yes	no	unknown	
1	44	technician	single	secondary	no	29	yes	no	unknown	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	
4	33	unknown	single	unknown	no	1	no	no	unknown	

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

Out[6]:

	age	job	marital	education	default	balance	housing	loan	conta
45206	51	technician	married	tertiary	no	825	no	no	cellu
45207	71	retired	divorced	primary	no	1729	no	no	cellu
45208	72	retired	married	secondary	no	5715	no	no	cellu
45209	57	blue-collar	married	secondary	no	668	no	no	telepho
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellu

In [7]:
```
df.shape
```

Out[7]:
```
(45211, 17)
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

In [9]:
```
df.dtypes
```

Out[9]:
```
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
```

In [10]:
```
df.dtypes.value_counts()
```

Out[10]:
```
object    10
int64      7
Name: count, dtype: int64
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  deposit    45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
df.isna().sum()
```

Out[13]:
```
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
```

In [14]:
```
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'], dty
pe='object')
```

In [15]:
```
df.describe()
```

Out[15]:

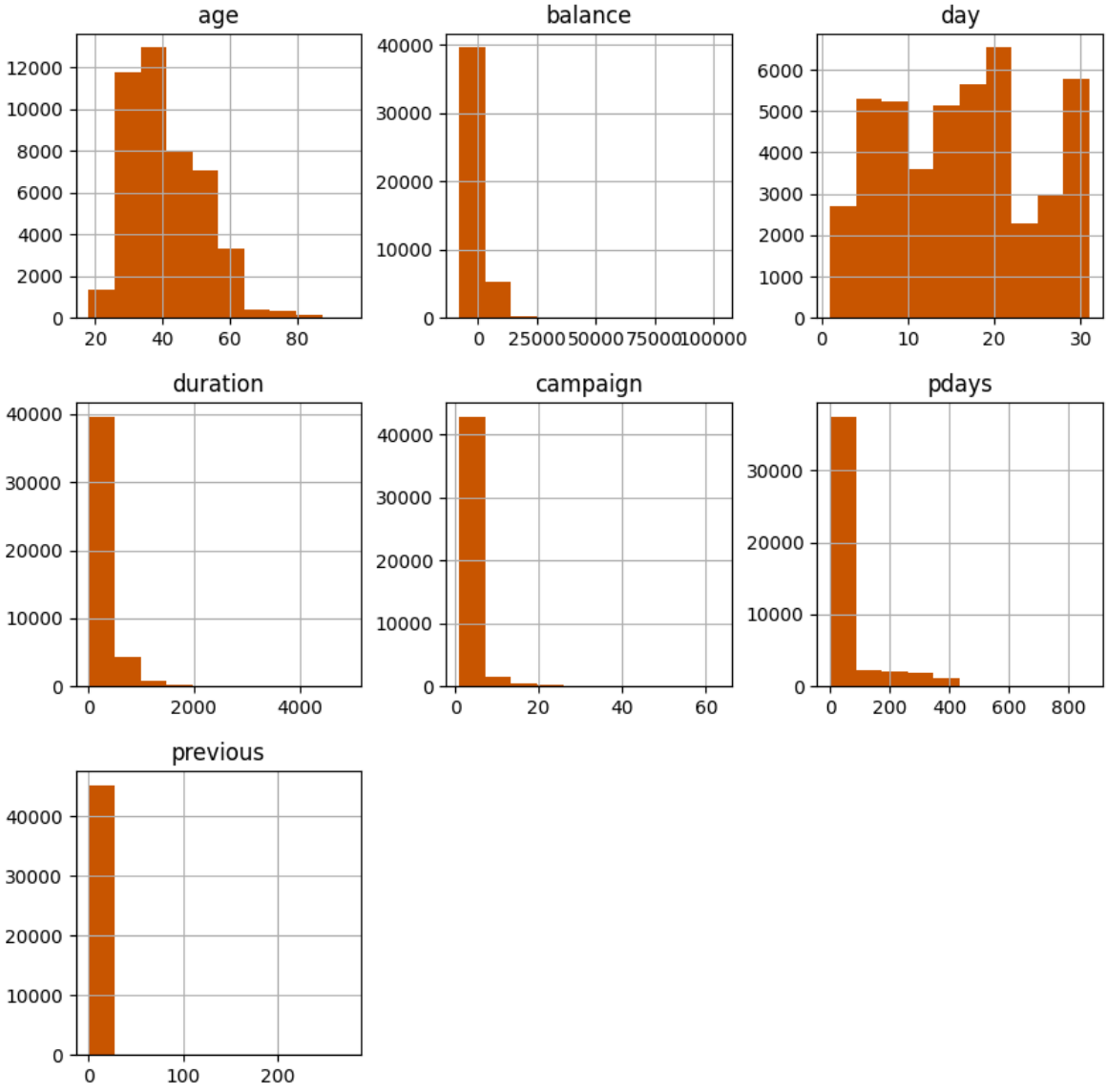
	age	balance	day	duration	campaign	p
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.00
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.19
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.12
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.00
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.00
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.00
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.00
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.00

In [16]:
```
df.describe(include='object')
```

Out[16]:

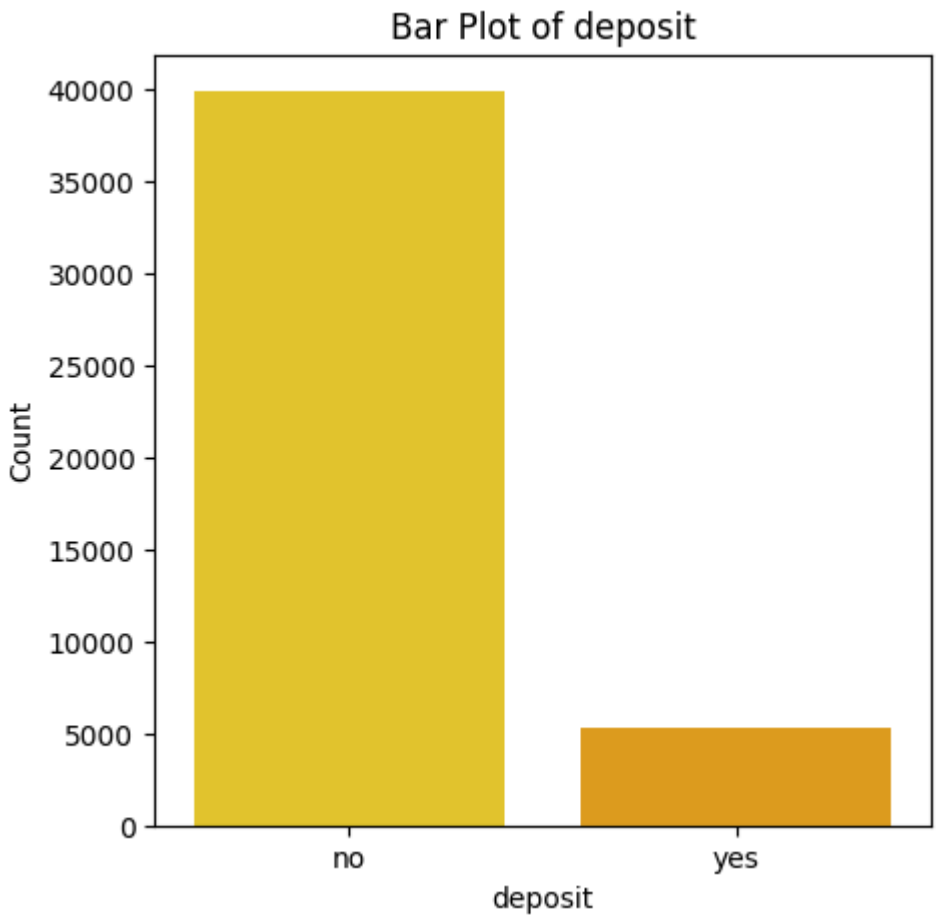
	job	marital	education	default	housing	loan	contact	month	poutcom
count	45211	45211	45211	45211	45211	45211	45211	45211	4521
unique	12	3	4	2	2	2	3	12	
top	blue-collar	married	secondary	no	yes	no	cellular	may	unknow
freq	9732	27214	23202	44396	25130	37967	29285	13766	3695

In [17]:
```
df.hist(figsize=(10,10),color='#cc5500')
plt.show()
```

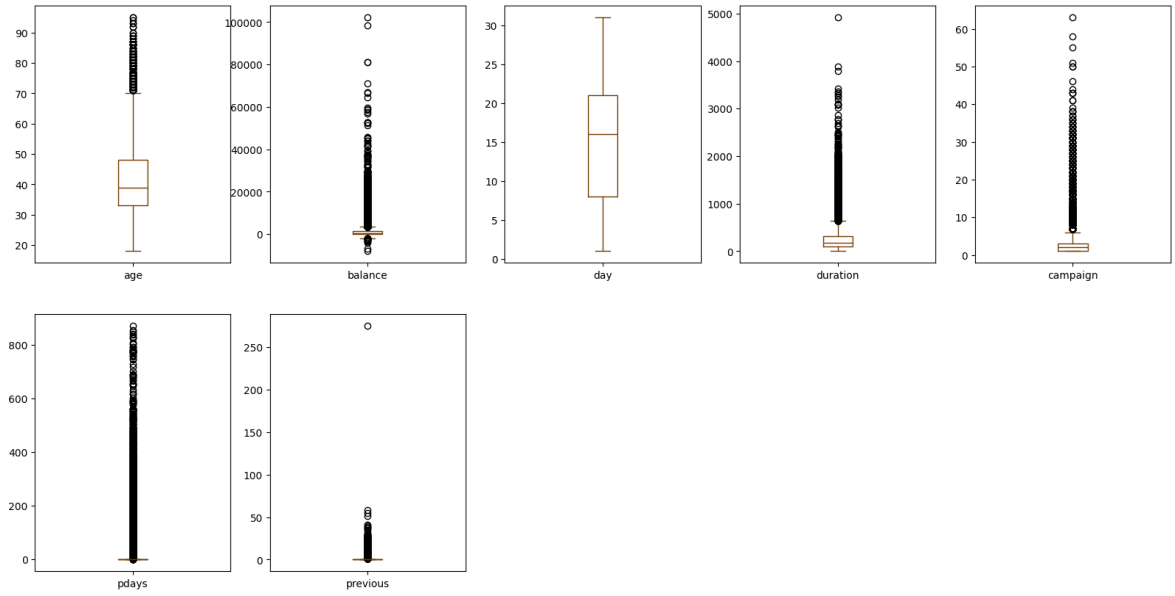


In [19]:
```
for feature in cat_cols:

    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')
```

Out[19]:
```
Text(0, 0.5, 'Count')
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
<Figure size 500x500 with 0 Axes>
```

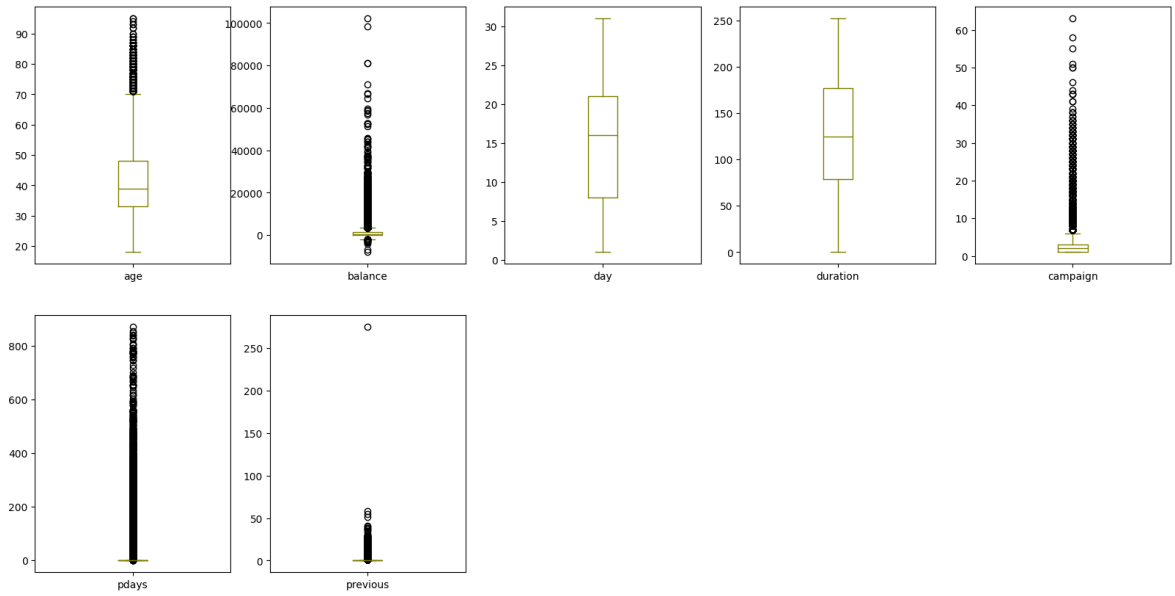


In [20]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00'
plt.show()
```



In [23]:
```
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 < up
```

In [25]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#808000')
plt.show()
```



In [28]:
```
corr = df.corr()
          print(corr)
           corr = corr[abs(corr)>=0.90]
            sns.heatmap(corr,annot=True,cmap='Set3',linewidths=0.2)
             plt.show()
```

```
  Cell In[28], line 2
    print(corr)
    ^
IndentationError: unexpected indent
```

In [30]:
```
high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

In [31]:
```
df1 = df.copy()
df1.columns
```

Out[31]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

In [34]:
```
df1.shape
```

Out[34]:
```
(45211, 17)
```

In [35]:
```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

Out[35]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
0	40	4	1	2	0	3036	1	0	2	4	
1	26	9	2	1	0	945	1	0	2	4	
2	15	2	1	1	0	918	1	1	2	4	
3	29	1	1	3	0	2420	1	0	2	4	
4	15	11	2	3	0	917	0	0	2	4	
...	...	...	...	...	...	...	...	...	...	...	
45206	33	9	1	2	0	1741	0	0	0	16	
45207	53	5	0	0	0	2639	0	0	0	16	
45208	54	5	1	1	0	5455	0	0	0	16	
45209	39	1	1	1	0	1584	0	0	1	16	
45210	19	2	1	1	0	3779	0	0	0	16	

45211 rows × 17 columns

In [36]:
```
df_encoded['deposit'].value_counts()
```

Out[36]:
```
deposit
0    39922
1     5289
Name: count, dtype: int64
```

In [37]:
```
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))
```

```
(45211, 16)
(45211,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [39]:
```
from sklearn.model_selection import train_test_split
print(4119*0.25)
```

```
1029.75
```

In [41]:
```
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)
```

```
(33908, 16)
(11303, 16)
(33908,)
(11303,)
```

In [42]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_sco

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 [43]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini',max_depth=5,min_samples_split=10)
dt.fit(x_train,y_train)
```

Out[43]:

DecisionTreeClassifier

DecisionTreeClassifier(max_depth=5, min_samples_split=10)

In [44]:
```
mscore(dt)
```

```
Training Score 0.8914120561519405
Testing Score 0.8882597540475979
```

In [45]:
```
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

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

In [46]:
```
eval_model(y_test,ypred_dt)
```

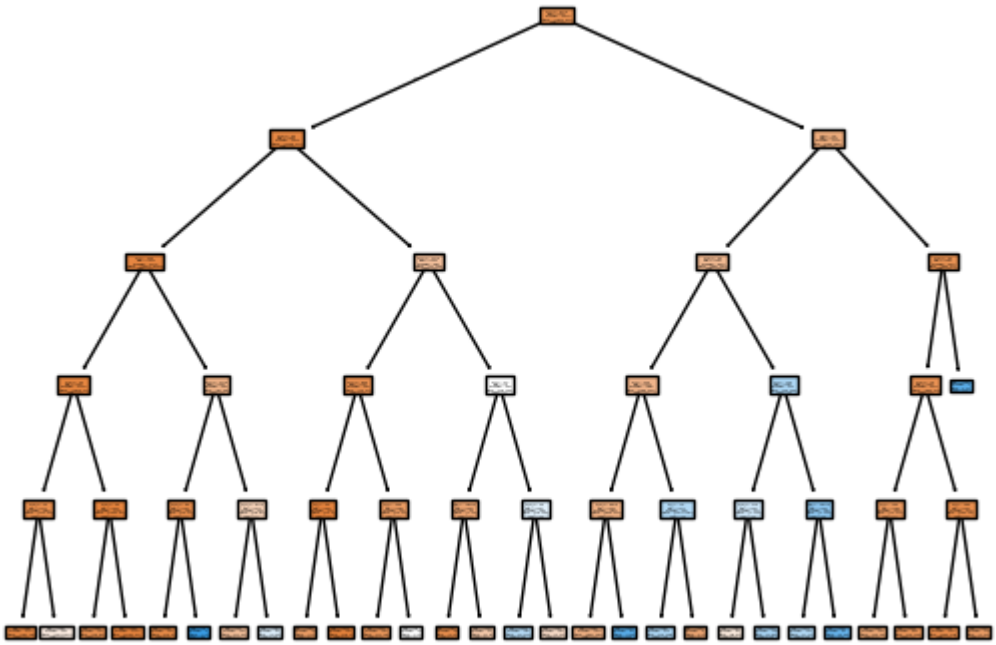
```
Accuracy_Score 0.8882597540475979
Confusion Matrix
 [[9860  138]
 [1125  180]]
Classification Report
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      9998
           1       0.57      0.14      0.22      1305

    accuracy                           0.89     11303
   macro avg       0.73      0.56      0.58     11303
weighted avg       0.86      0.89      0.86     11303
```

In [47]:
```
from sklearn.tree import plot_tree
```

In [49]:
```
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']
```

In [50]:
```
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



In [51]:
```
dt1 = DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=
dt1.fit(x_train,y_train)
```

Out[51]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_sp
lit=15)
```

In [52]:
```
mscore(dt1)
```

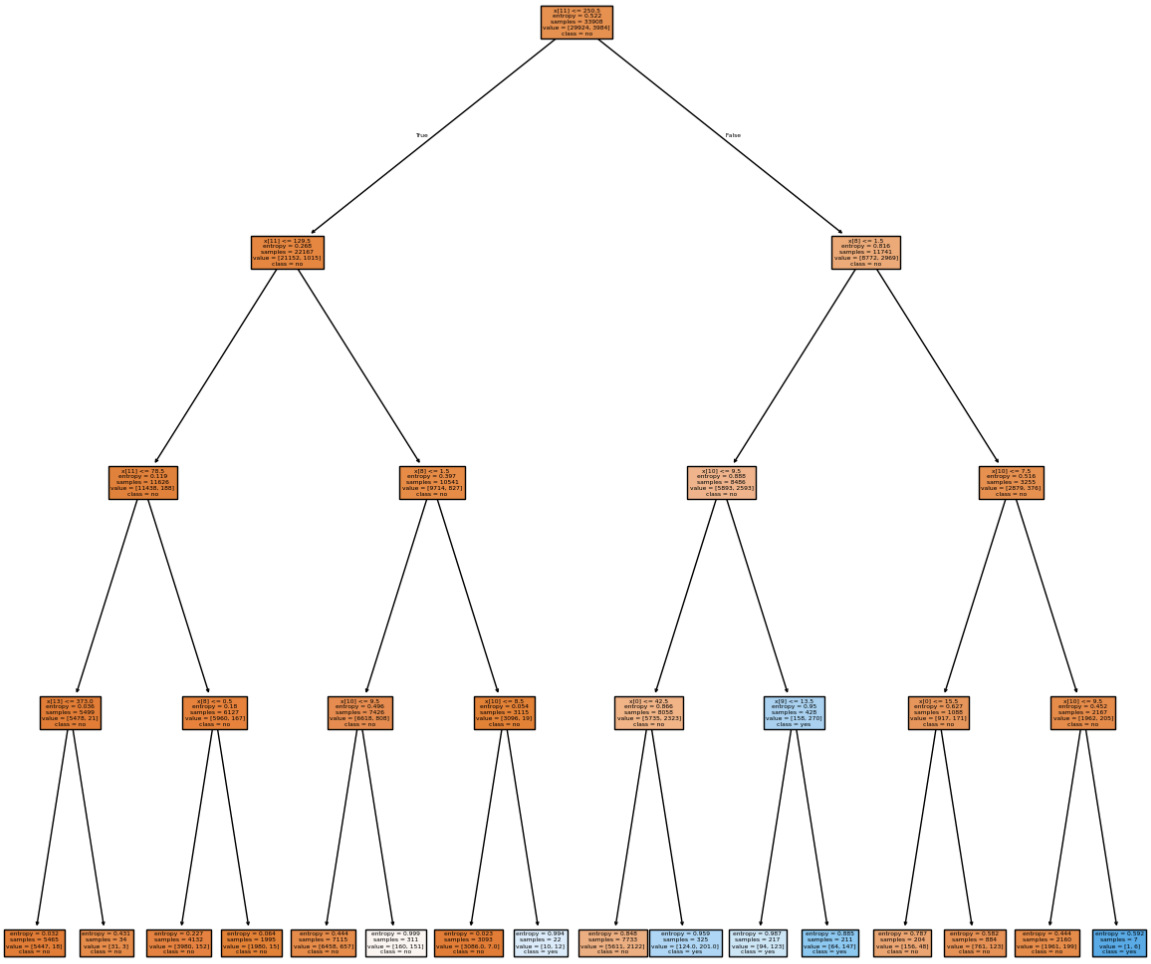
```
Training Score 0.8882859502182375
Testing Score 0.8885251703087675
```

In [53]:
```
ypred_dt1 = dt1.predict(x_test)
```

In [54]:
```
eval_model(y_test,ypred_dt1)
```

```
Accuracy_Score 0.8885251703087675
Confusion Matrix
 [[9906   92]
 [1168  137]]
Classification Report
              precision    recall  f1-score   support

           0       0.89      0.99      0.94      9998
           1       0.60      0.10      0.18      1305

    accuracy                           0.89     11303
   macro avg       0.75      0.55      0.56     11303
weighted avg       0.86      0.89      0.85     11303
```

In [55]:
```
plt.figure(figsize=(15,15))
plot_tree(dt1,class_names=cn,filled=True)
plt.show()
```



In [ ]:
```

```