TASK - 3

Importing necessary libraries

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
```

Reading the dataset

In [2]:
```
df=pd.read_csv("bank-additional.csv",delimiter=';')
# delimiter is used to seperate the columns

df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previ
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	

5 rows × 21 columns

Some information about the dataset

In [3]:
```
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 [4]:
```
df.describe()
```

Out[4]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704	-40.499102	3.621356
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349	4.594578	1.733591
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.635000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.334000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000	-26.900000	5.045000

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

Out[5]:
```
(4119, 21)
```

In [6]:
```
df.columns
```

Out[6]:
```
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')
```

Checking for the datatypes

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

Out[7]:
```
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 [8]:
```
df.dtypes.value_counts()
```

Out[8]:
```
object     11
int64       5
float64     5
Name: count, dtype: int64
```

Checking for the duplicate values

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

Out[9]:
```
np.int64(0)
```

Checking for null values

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	cons.price.idx	cons.conf.idx	euribor3m
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704	-40.499102	3.621356
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349	4.594578	1.733591
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.635000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.334000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000	-26.900000	5.045000

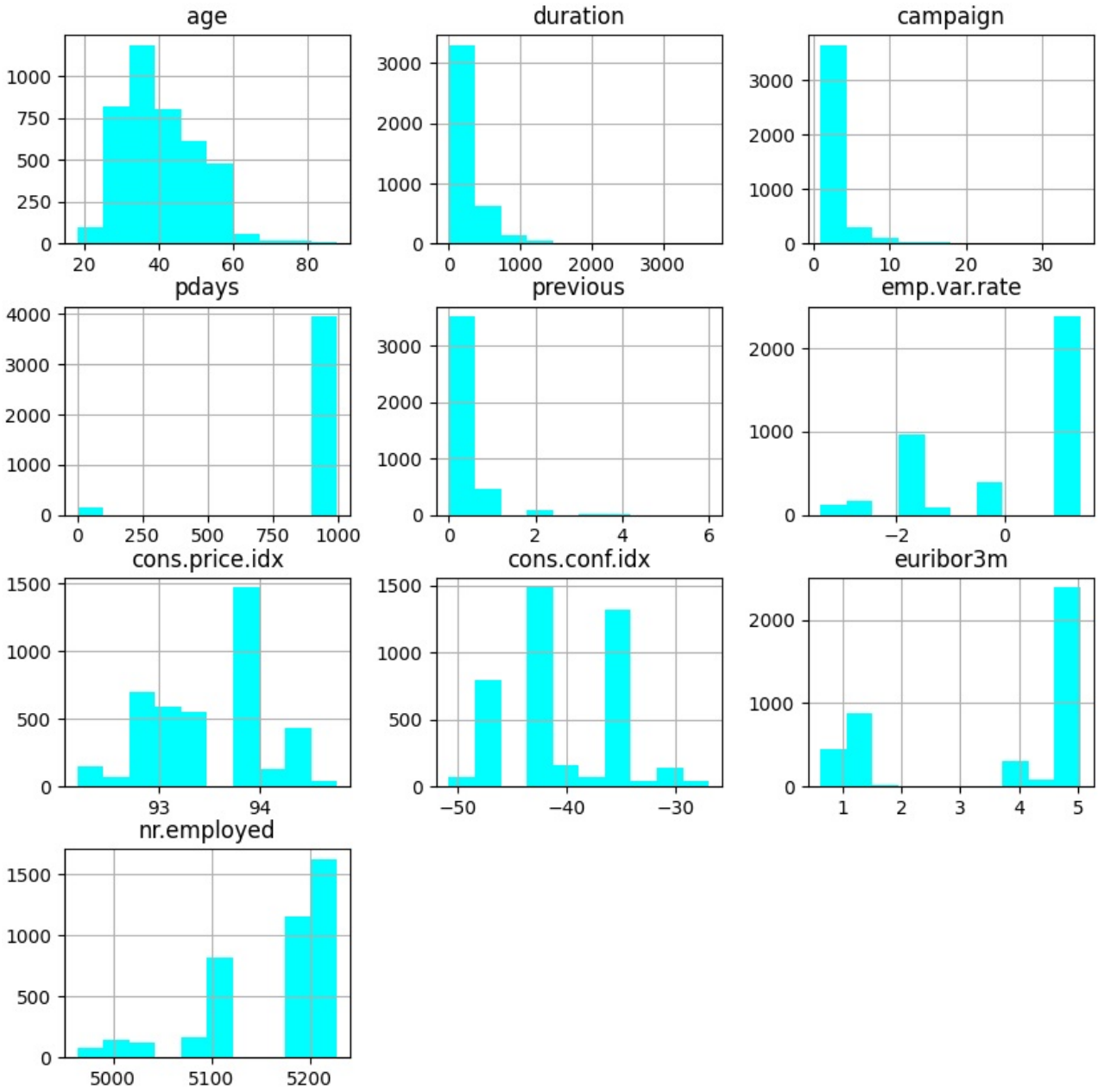
In [13]:
```
df.describe(include='object')
```

Out[13]:

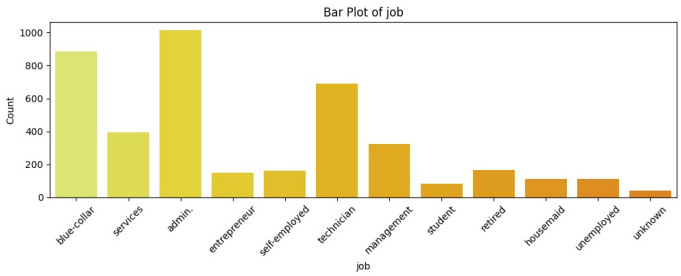
	job	marital	education	default	housing	loan	contact	month	day_of_week	poutcome	deposit
count	4119	4119	4119	4119	4119	4119	4119	4119	4119	4119	4119
unique	12	4	8	3	3	3	2	10	5	3	2
top	admin.	married	university.degree	no	yes	no	cellular	may	thu	nonexistent	no
freq	1012	2509	1264	3315	2175	3349	2652	1378	860	3523	3668

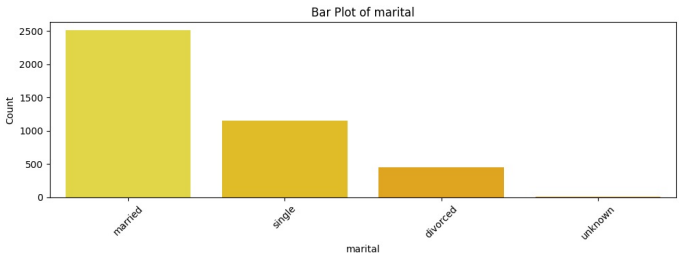
Visualisation

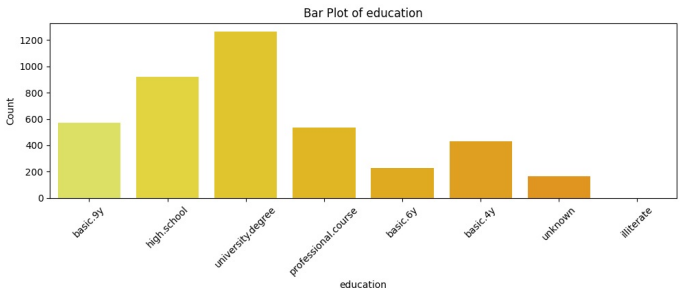
In [14]:
```
df.hist(figsize=(10,10),color='#00FFFF')
plt.show()
```

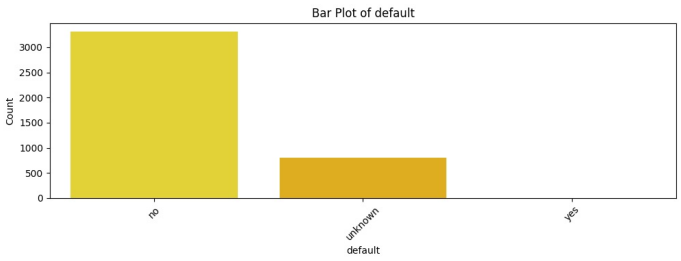


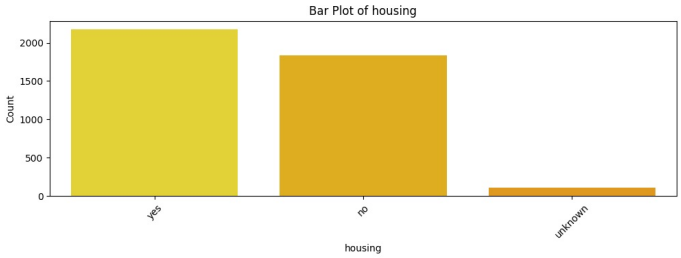
In [15]:
```
# Calculate the number of rows and columns for subplots
num_plots = len(cat_cols)
num_rows = (num_plots + 1) // 2  # Add 1 and divide by 2 to round up for odd numbers
num_cols = 2

# Create a new figure
plt.figure(figsize=(20, 25))  # Adjust the figure size as needed

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

# Adjust layout to prevent overlap of subplots
plt.tight_layout()
plt.show()
```

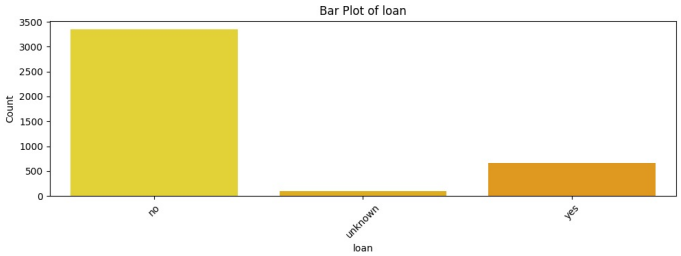


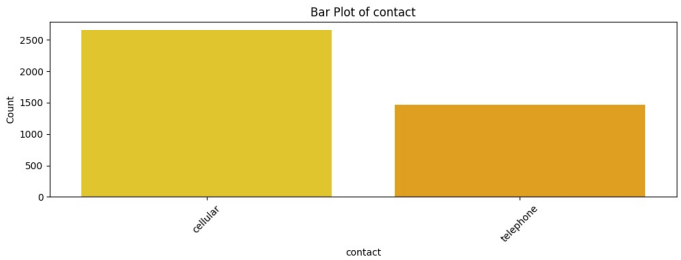


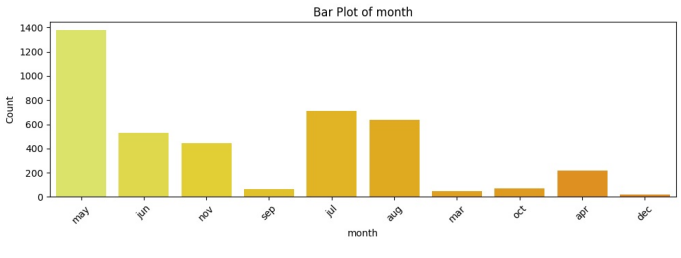


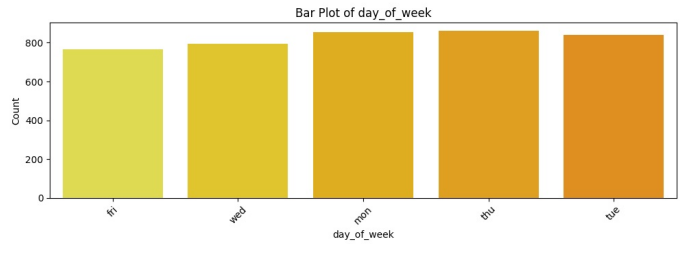


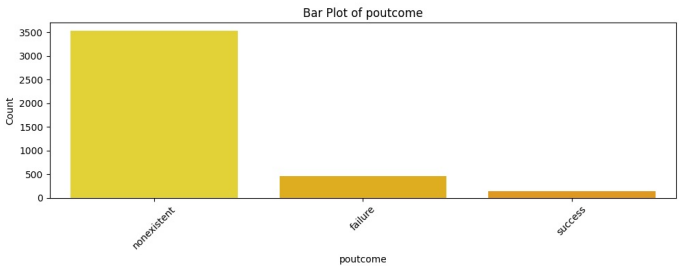


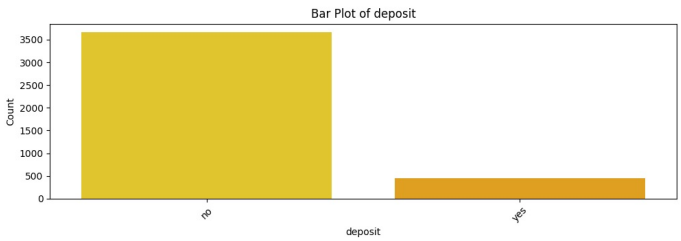












In [16]:
```
high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

In [17]:
```
df1 = df.copy()
df1.columns
```

Out[17]:
```
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 [18]:
```
df1.drop(high_corr_cols,inplace=True,axis=1)  # axis=1 indicates columns
df1.columns
```

Out[18]:
```
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 [19]:
```
%pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\komal\appdata\local\programs\python\python314\lib\site-p
ackages (1.8.0)
Requirement already satisfied: numpy>=1.24.1 in c:\users\komal\appdata\local\programs\python\python314\lib\site-
packages (from scikit-learn) (2.4.0)
Requirement already satisfied: scipy>=1.10.0 in c:\users\komal\appdata\local\programs\python\python314\lib\site-
packages (from scikit-learn) (1.16.3)
Requirement already satisfied: joblib>=1.3.0 in c:\users\komal\appdata\local\programs\python\python314\lib\site-
packages (from scikit-learn) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.2.0 in c:\users\komal\appdata\local\programs\python\python314\li
b\site-packages (from scikit-learn) (3.6.0)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:
```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

Out[20]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	p
0	12	1	1	2	0	2	0	0	6	0	474	1	20	0	
1	21	7	2	3	0	0	0	1	6	0	343	3	20	0	
2	7	7	1	3	0	2	0	1	4	4	224	0	20	0	
3	20	7	1	2	0	1	1	1	4	0	14	2	20	0	
4	29	0	1	6	0	2	0	0	7	1	55	0	20	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4114	12	0	1	1	0	2	2	0	3	2	50	0	20	0	
4115	21	0	1	3	0	2	0	1	3	0	216	0	20	0	
4116	9	8	2	3	0	0	0	0	6	1	61	1	20	1	
4117	40	0	1	3	0	0	0	0	1	0	510	0	20	0	
4118	16	4	2	3	0	2	0	0	7	4	172	0	20	0	

4119 rows × 18 columns

In [21]:
```
df_encoded['deposit'].value_counts()
```

Out[21]:
```
deposit
0    3668
1     451
Name: count, dtype: int64
```

In [22]:
```
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 [23]:
```
from sklearn.model_selection import train_test_split
```

In [24]:
```
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,)
```

In [25]:
```
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 [26]:
```
from sklearn.tree import DecisionTreeClassifier

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

Out[26]:

▾ DecisionTreeClassifier ⓘ ⓘ

▶ Parameters

In [27]:
```
mscore(dt)
```

```
Training Score 0.923276141146002
Testing Score 0.9116504854368932
```

In [28]:
```
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

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

In [29]:
```
eval_model(y_test,ypred_dt)
```

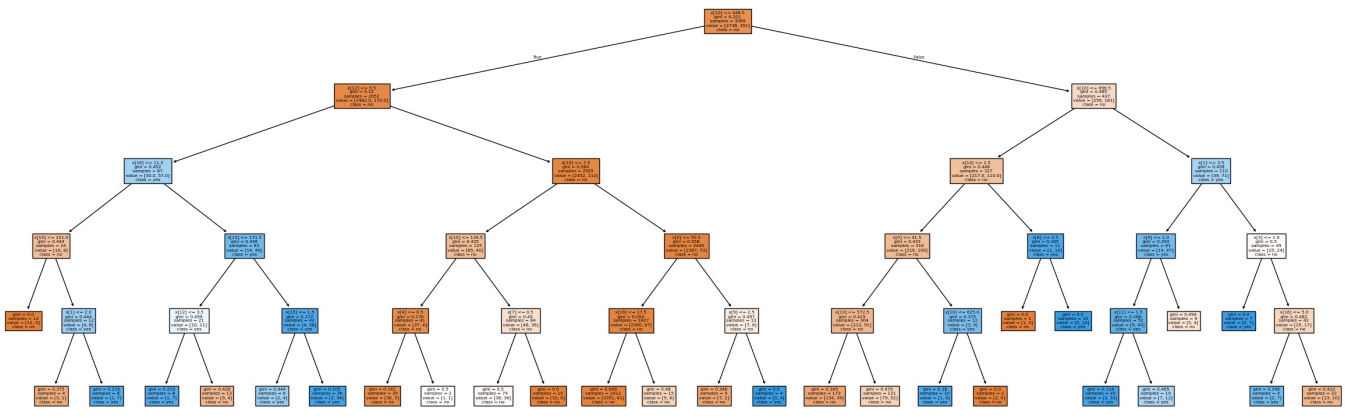
```
Accuracy_Score 0.9116504854368932
Confusion Matrix
 [[913  17]
 [ 74  26]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       930
           1       0.60      0.26      0.36       100

    accuracy                           0.91      1030
   macro avg       0.76      0.62      0.66      1030
weighted avg       0.89      0.91      0.90      1030
```

In [30]:
```
from sklearn.tree import plot_tree
```

In [31]:
```
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 [32]:
```
plt.figure(figsize=(30,10))
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



In [33]:
```
dt1 = DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=15)
```

```
dt1.fit(x_train,y_train)
```

Out[33]:

▾ DecisionTreeClassifier ⓘ ⓘ

▶ Parameters

In [34]:
```
mscore(dt1)
```

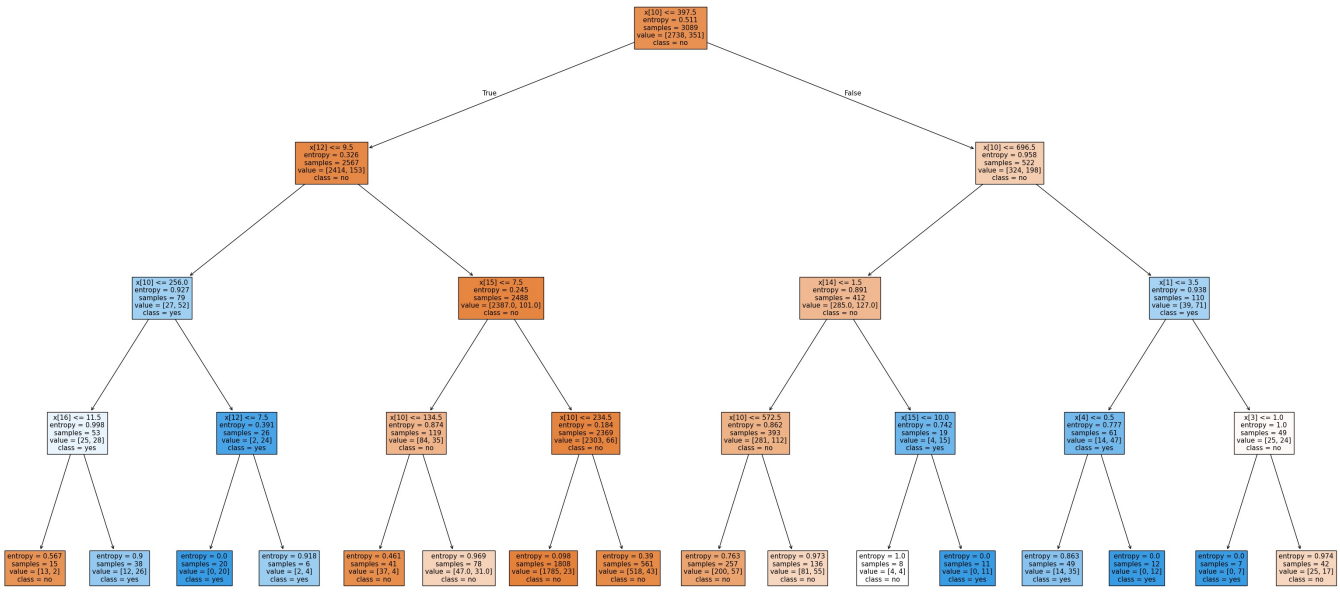
```
Training Score 0.9145354483651668
Testing Score 0.916504854368932
```

In [35]:
```
ypred_dt1 = dt1.predict(x_test)
```

In [36]:
```
eval_model(y_test,ypred_dt1)
```

```
Accuracy_Score 0.916504854368932
Confusion Matrix
 [[912  18]
 [ 68  32]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       930
           1       0.64      0.32      0.43       100

    accuracy                           0.92      1030
   macro avg       0.79      0.65      0.69      1030
weighted avg       0.90      0.92      0.90      1030
```

In [37]:
```
plt.figure(figsize=(40,20))
plot_tree(dt1,class_names=cn,filled=True)
plt.show()
```



In [ ]:
```
```