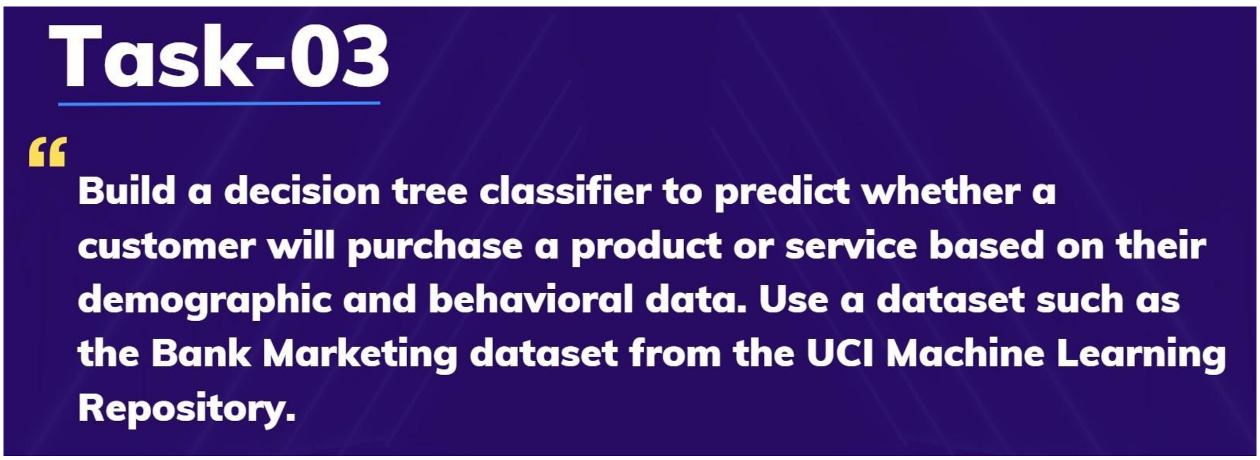
Prodigy Infotech Internship Task by Chaitanya gadekar

Task 3 : Decision Tree Classification on Bank Marketing Dataset

Problem Statement



About the Dataset :

The Dataset we are going to use in this task is Bank Marketing Dataset which is taken from UCI Machine Learning Repository.The dataset here is the 10% sample of the Original Bank Marketing Dataset. The dataset contains 20 features and 1 label. The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

Features of the Dataset:

Input variables:

bank client data:

1. age (numeric)
2. job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3. marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4. education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5. default: has credit in default? (categorical: 'no','yes','unknown')
6. housing: has housing loan? (categorical: 'no','yes','unknown')
7. loan: has personal loan? (categorical: 'no','yes','unknown') ## related with the last contact of the current campaign:
8. contact: contact communication type (categorical: 'cellular','telephone')
9. month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10. day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11. duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model. ## other attributes:
12. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14. previous: number of contacts performed before this campaign and for this client (numeric)
15. poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success') ## social and economic context attributes
16. emp.var.rate: employment variation rate - quarterly indicator (numeric)
17. cons.price.idx: consumer price index - monthly indicator (numeric)
18. cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19. euribor3m: euribor 3 month rate - daily indicator (numeric)
20. nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

1. y - has the client subscribed a term deposit? (binary: 'yes','no')

The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

Importing Libraries

```
import pandas as pd
# pandas is aliased as pd
import numpy as np
# numpy is aliased as np
import matplotlib.pyplot as plt
```

```
# pyplot is aliased as plt
import seaborn as sns
# seaborn is aliased as sns

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

Loading the Dataset

```
df = pd.read_csv('bank-additional.csv',delimiter=';')
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

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

     contact month day_of_week ...  campaign pdays  previous
poutcome \
0   cellular   may         fri  ...         2   999         0
nonexistent
1  telephone   may         fri  ...         4   999         0
nonexistent
2  telephone   jun         wed  ...         1   999         0
nonexistent
3  telephone   jun         fri  ...         3   999         0
nonexistent
4   cellular   nov         mon  ...         1   999         0
nonexistent

  emp.var.rate cons.price.idx cons.conf.idx euribor3m nr.employed
deposit
0         -1.8         92.893          -46.2     1.313       5099.1
no
1          1.1         93.994          -36.4     4.855       5191.0
no
2          1.4         94.465          -41.8     4.962       5228.1
```

```
no
3           1.4          94.465          -41.8      4.959        5228.1
no
4          -0.1          93.200          -42.0      4.191        5195.8
no

[5 rows x 21 columns]
```

```python
# Showing first 5 rows
df.head()
```

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

     contact month day_of_week  ...  campaign  pdays  previous
poutcome  \
0   cellular   may         fri  ...         2    999         0
nonexistent
1  telephone   may         fri  ...         4    999         0
nonexistent
2  telephone   jun         wed  ...         1    999         0
nonexistent
3  telephone   jun         fri  ...         3    999         0
nonexistent
4   cellular   nov         mon  ...         1    999         0
nonexistent

  emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed
deposit
0          -1.8          92.893          -46.2      1.313        5099.1
no
1           1.1          93.994          -36.4      4.855        5191.0
no
2           1.4          94.465          -41.8      4.962        5228.1
no
3           1.4          94.465          -41.8      4.959        5228.1
no
4          -0.1          93.200          -42.0      4.191        5195.8
no
```

```
[5 rows x 21 columns]

# showing last 5 rows
df.tail()

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

     month day_of_week  ...  campaign  pdays  previous     poutcome  \
4114   jul         thu  ...         1    999         0  nonexistent
4115   jul         fri  ...         1    999         0  nonexistent
4116   may         mon  ...         2    999         1      failure
4117   aug         fri  ...         1    999         0  nonexistent
4118   nov         wed  ...         1    999         0  nonexistent

     emp.var.rate cons.price.idx cons.conf.idx euribor3m
nr.employed \
4114          1.4         93.918          -42.7      4.958
5228.1
4115          1.4         93.918          -42.7      4.959
5228.1
4116         -1.8         92.893          -46.2      1.354
5099.1
4117          1.4         93.444          -36.1      4.966
5228.1
4118         -0.1         93.200          -42.0      4.120
5195.8

      deposit
4114       no
4115       no
4116       no
4117       no
4118       no

[5 rows x 21 columns]
```

Basic Understanding of the Dataset

```
# showing dimensions of the dataset
df.shape
```

```
(4119, 21)
```

The dataset contains 4119 rows and 21 columns

```
# showing column names
df.columns
```

```
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 data types
df.dtypes
```

```
# checking for different data types
df.dtypes.value_counts()
```

```
float64      5
dtype: int64

# showing information about the dataset
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
```

From the above information we can conclude that -

1. The dataset has 21 columns and 4119 rows.
2. The dataset has 11 categorical columns
3. The dataset has 10 numerical columns
4. the dataset has no null values

Data Cleaning and Data Preprocessing

Handling Duplicated Values

```
# checking for duplicates
df.duplicated().sum()
```

```
0
```

Handling Null Values

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

There is no null values in the dataset

Extracting Numerical and Categorical Columns

```
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')
```

Descriptive Statistical Analysis

```
# For Numerical Columns
df.describe()
```

```
               age     duration     campaign        pdays     previous
\
count  4119.000000  4119.000000  4119.000000  4119.000000  4119.000000

mean     40.113620   256.788055     2.537266   960.422190     0.190337

std      10.313362   254.703736     2.568159   191.922786     0.541788

min      18.000000     0.000000     1.000000     0.000000     0.000000

25%      32.000000   103.000000     1.000000   999.000000     0.000000

50%      38.000000   181.000000     2.000000   999.000000     0.000000

75%      47.000000   317.000000     3.000000   999.000000     0.000000

max      88.000000  3643.000000    35.000000   999.000000     6.000000


       emp.var.rate  cons.price.idx  cons.conf.idx    euribor3m
nr.employed
count   4119.000000     4119.000000    4119.000000  4119.000000
4119.000000
mean       0.084972       93.579704     -40.499102     3.621356
5166.481695
std        1.563114        0.579349       4.594578     1.733591
73.667904
min       -3.400000       92.201000     -50.800000     0.635000
4963.600000
25%       -1.800000       93.075000     -42.700000     1.334000
5099.100000
50%        1.100000       93.749000     -41.800000     4.857000
5191.000000
75%        1.400000       93.994000     -36.400000     4.961000
5228.100000
max        1.400000       94.767000     -26.900000     5.045000
5228.100000
```

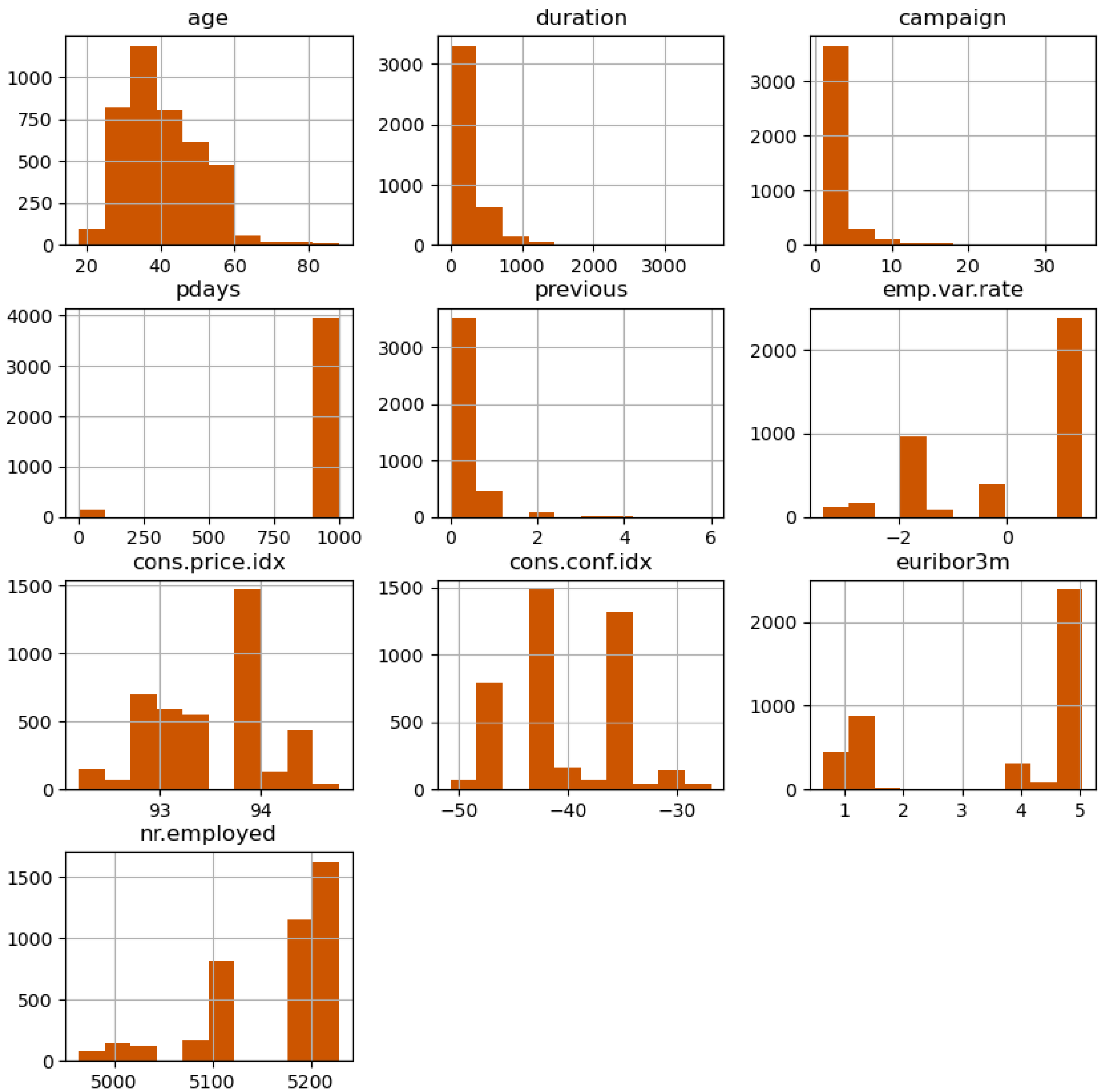
```python
# For Categorical columns
df.describe(include='object')
```

```
           job  marital          education default housing   loan
contact  \
count     4119     4119               4119    4119    4119   4119
4119
unique      12        4                  8       3       3      3
2
top     admin.  married  university.degree      no     yes     no
cellular
freq      1012     2509               1264    3315    2175   3349
2652

       month day_of_week     poutcome deposit
count   4119        4119         4119    4119
unique    10           5            3       2
top      may         thu  nonexistent      no
freq    1378         860         3523    3668
```

Data Visualization

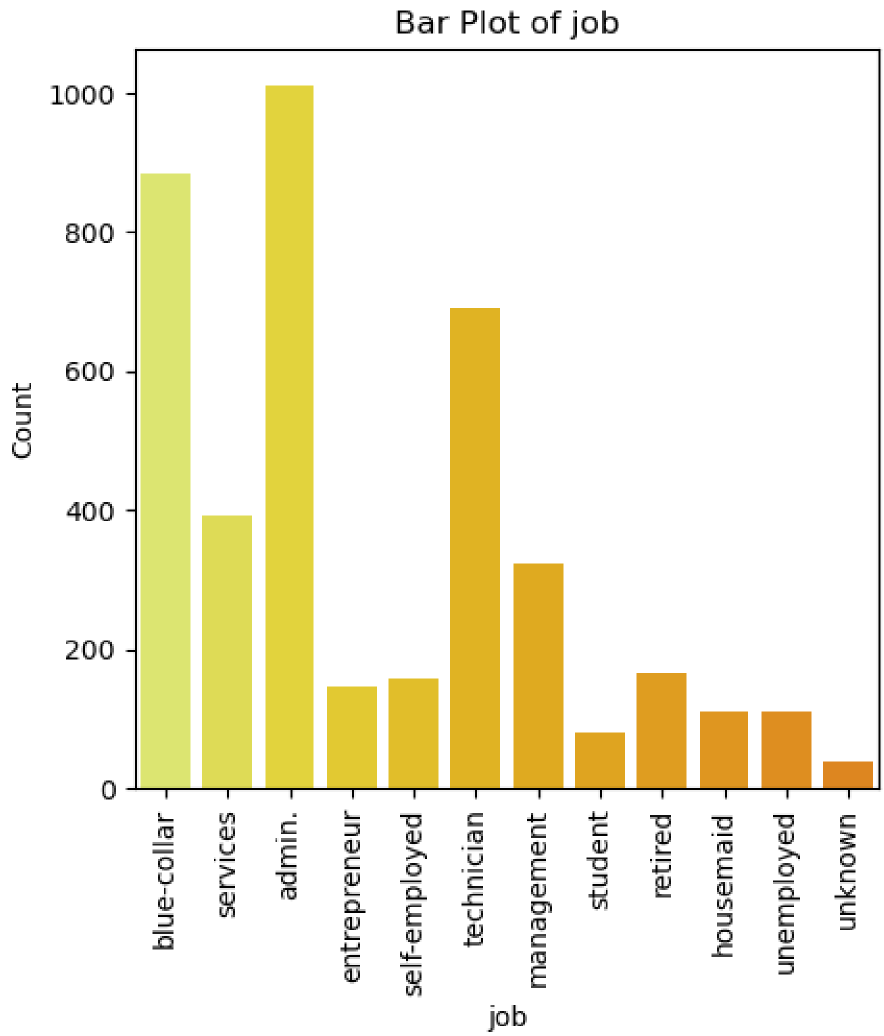
Visualizing Numerical columns using Histplot

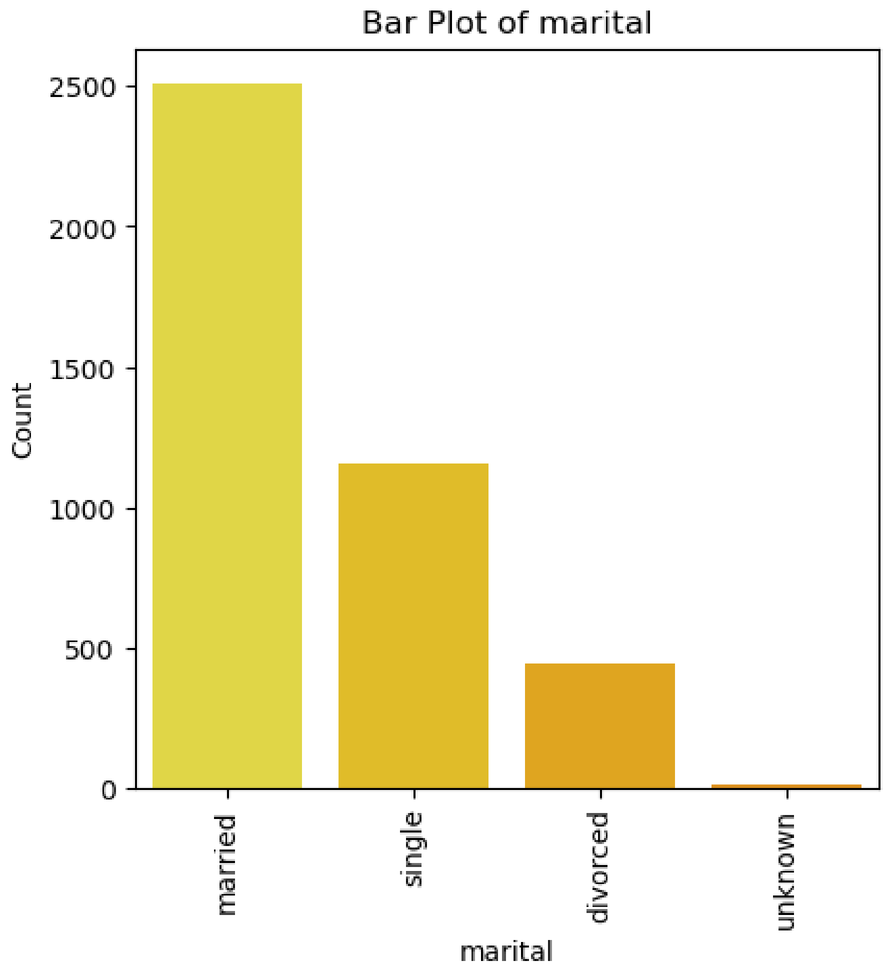
```python
df.hist(figsize=(10,10),color='#cc5500')
plt.show()
```

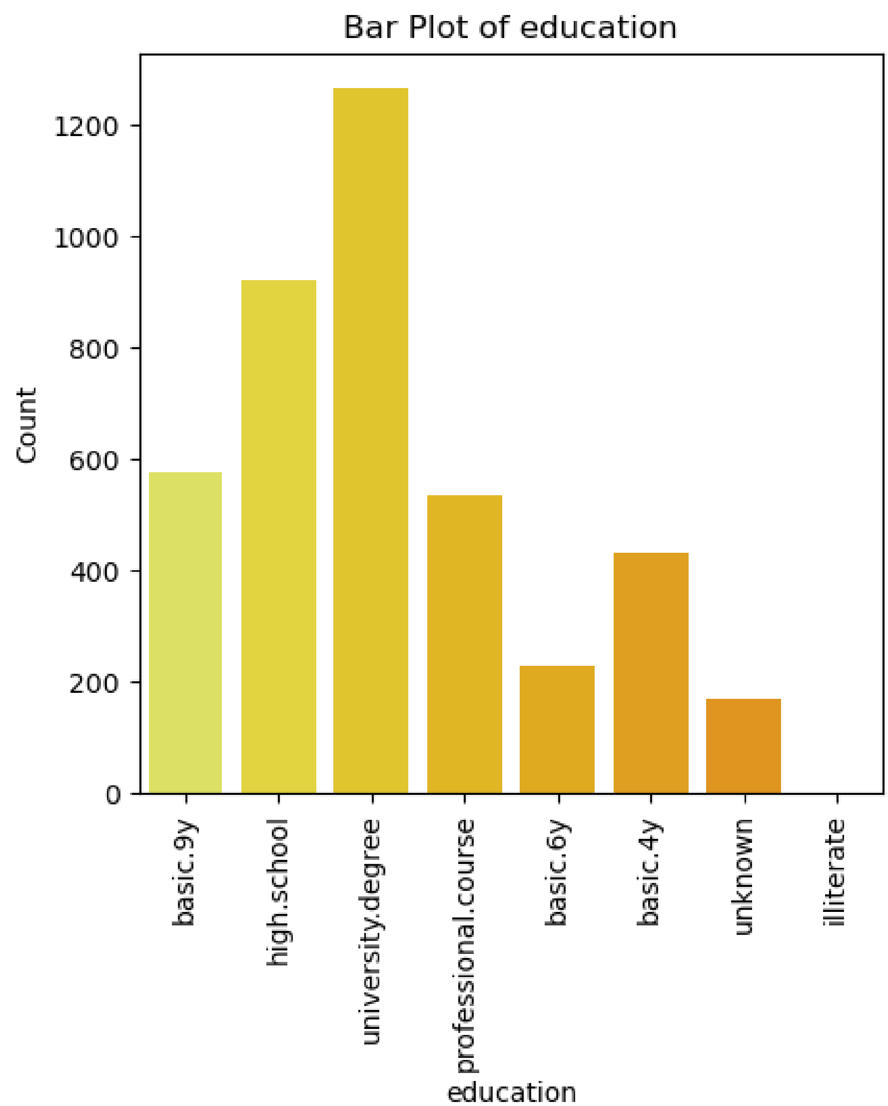


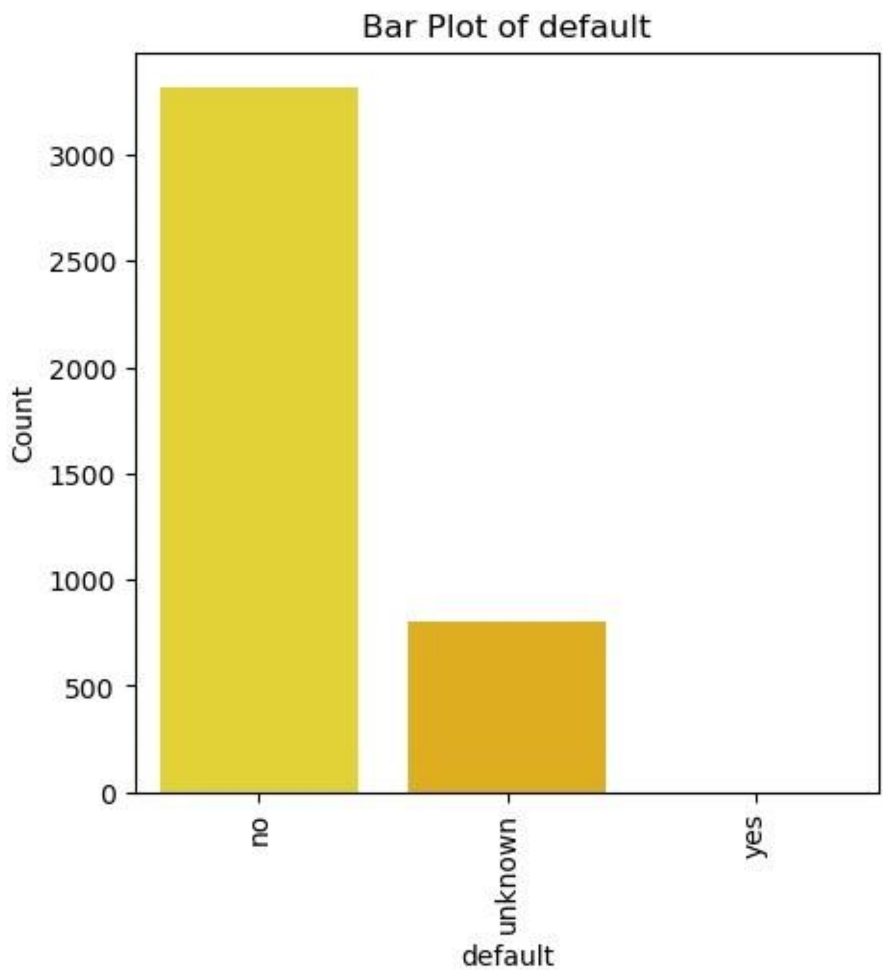
Visualizing Categorical columns using Barplot

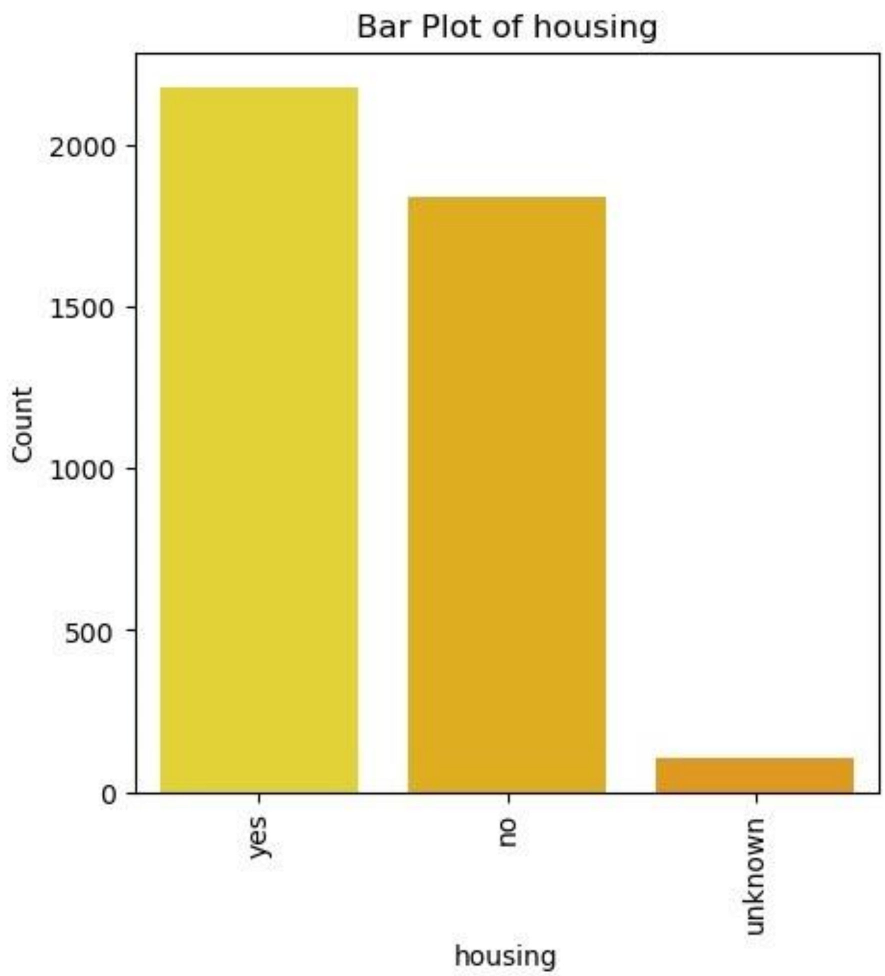
```
for feature in cat_cols:
    plt.figure(figsize=(5,5))  # Adjust the figure size as needed
    sns.countplot(x=feature, data=df, palette='Wistia')
    plt.title(f'Bar Plot of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=90)
    plt.show()
```

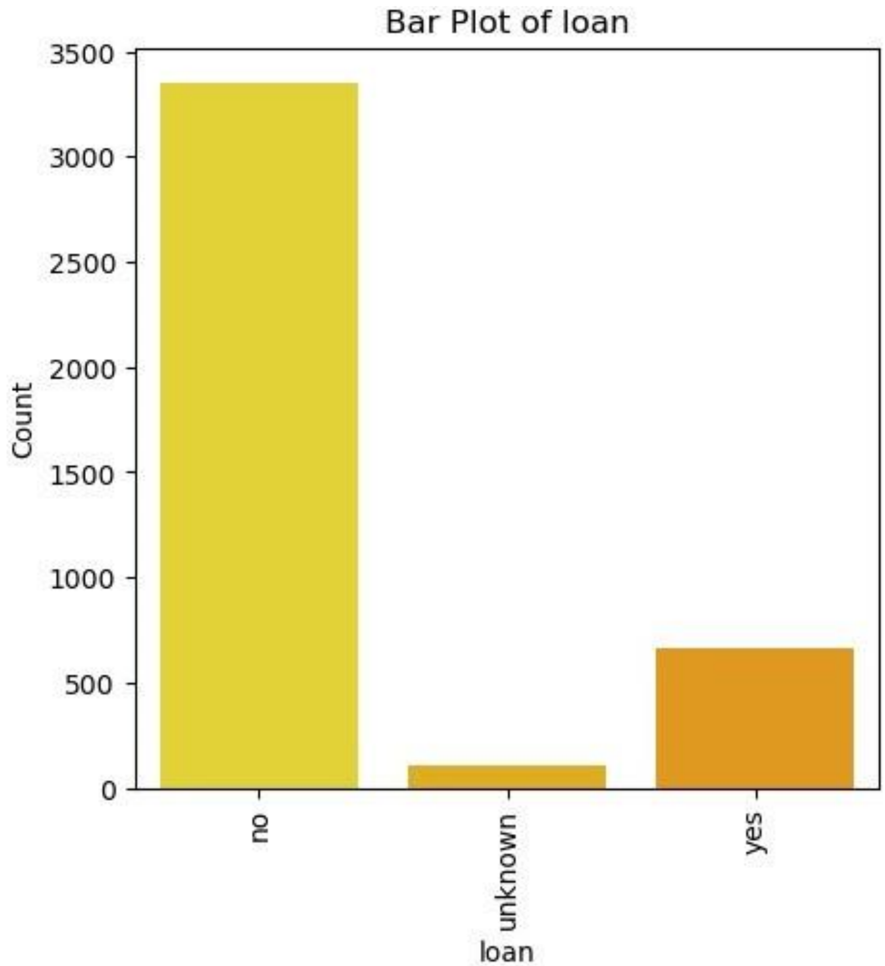

Bar Plot of job
Count
1000
800
600
400
200
0
blue-collar
services
admin.
entrepreneur
self-employed
technician
management
student
retired
housemaid
unemployed
unknown
job

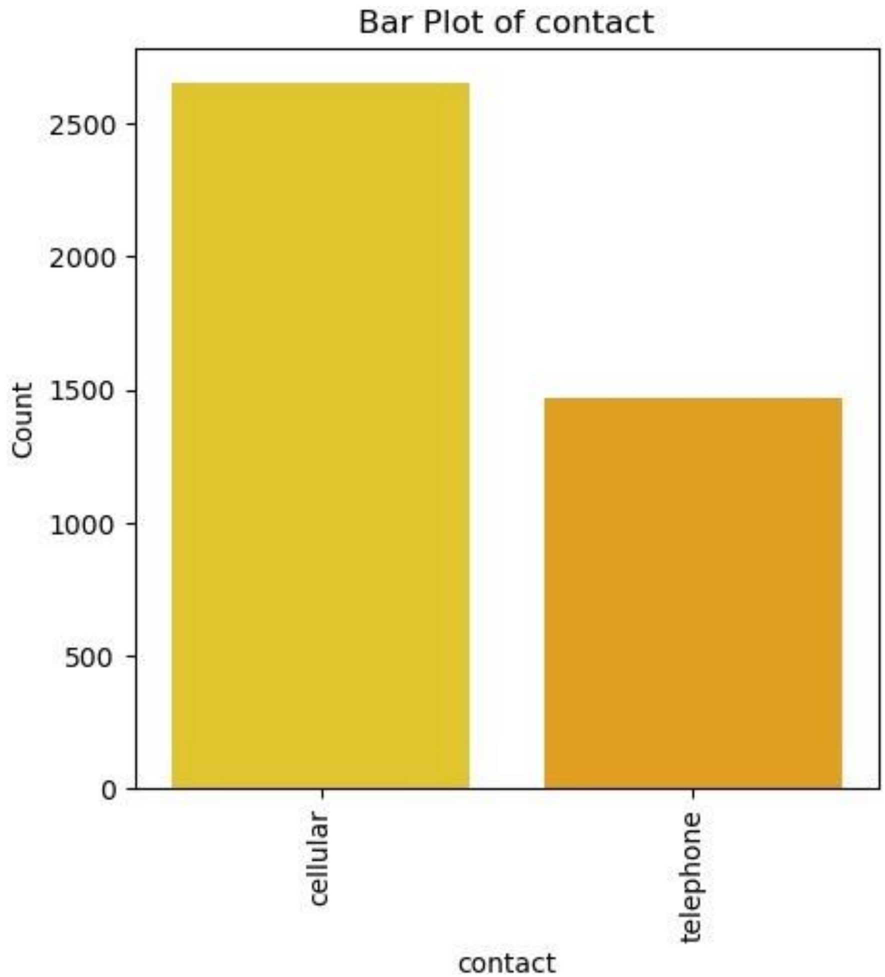

Bar Plot of marital
2500
2000
1500
1000
500
0
Count
married
single
divorced
unknown
marital

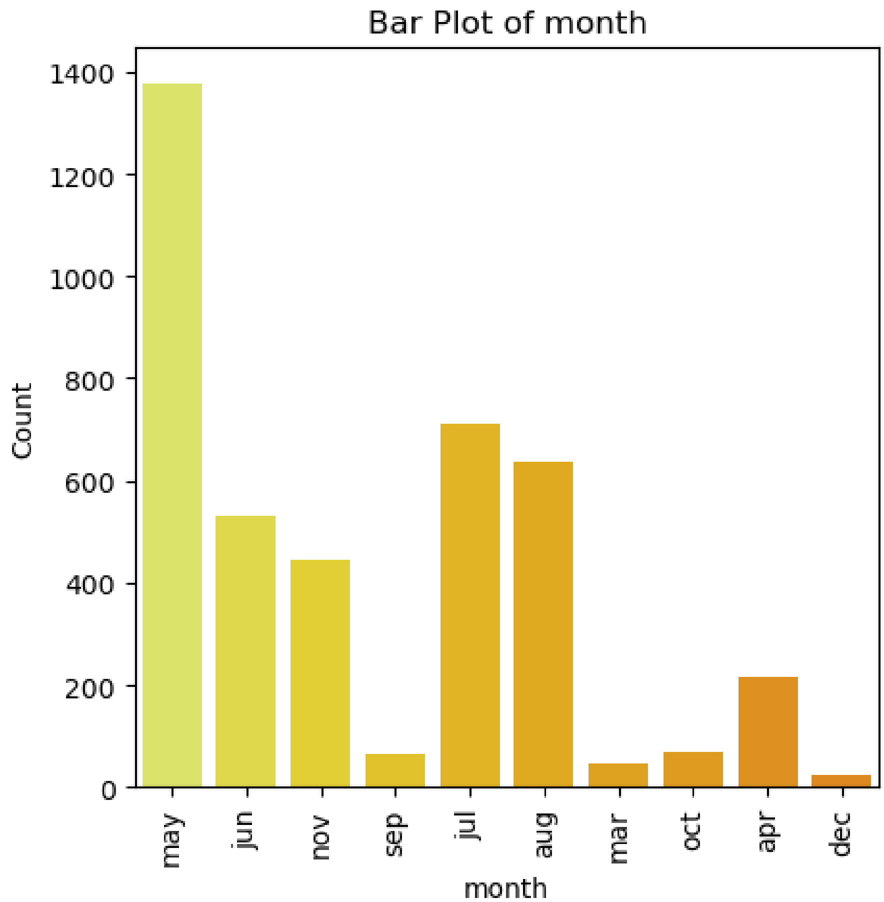

Bar Plot of education
Count
0
200
400
600
800
1000
1200
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate
education

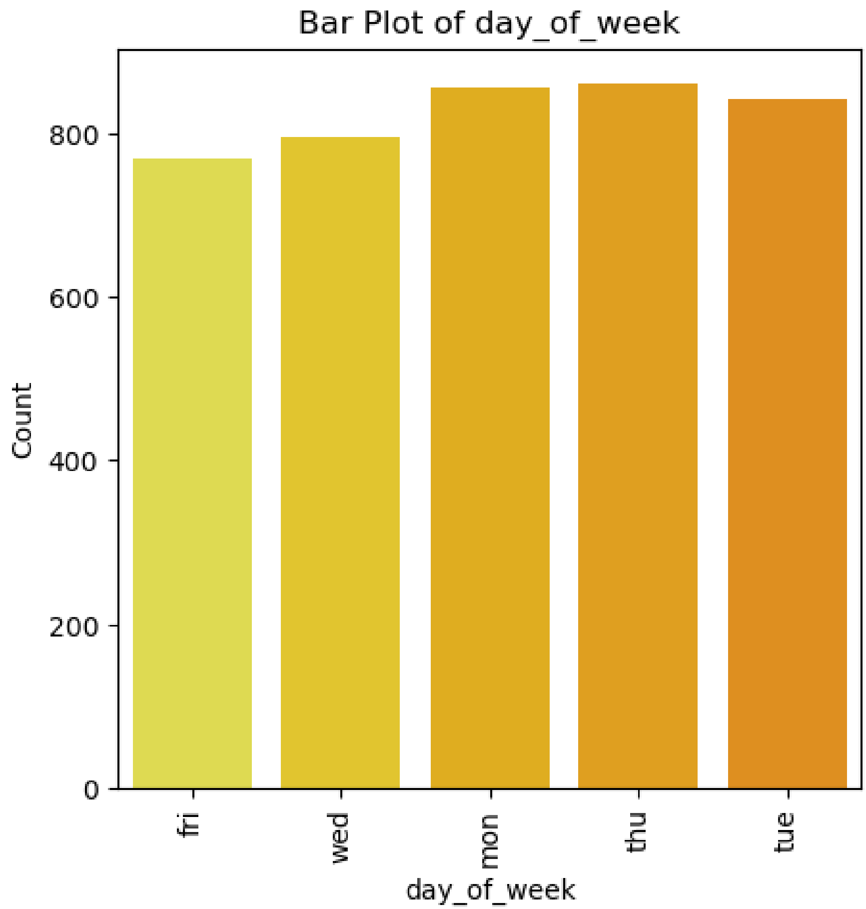

Bar Plot of default
Count
3000
2500
2000
1500
1000
500
0
no
unknown
yes
default

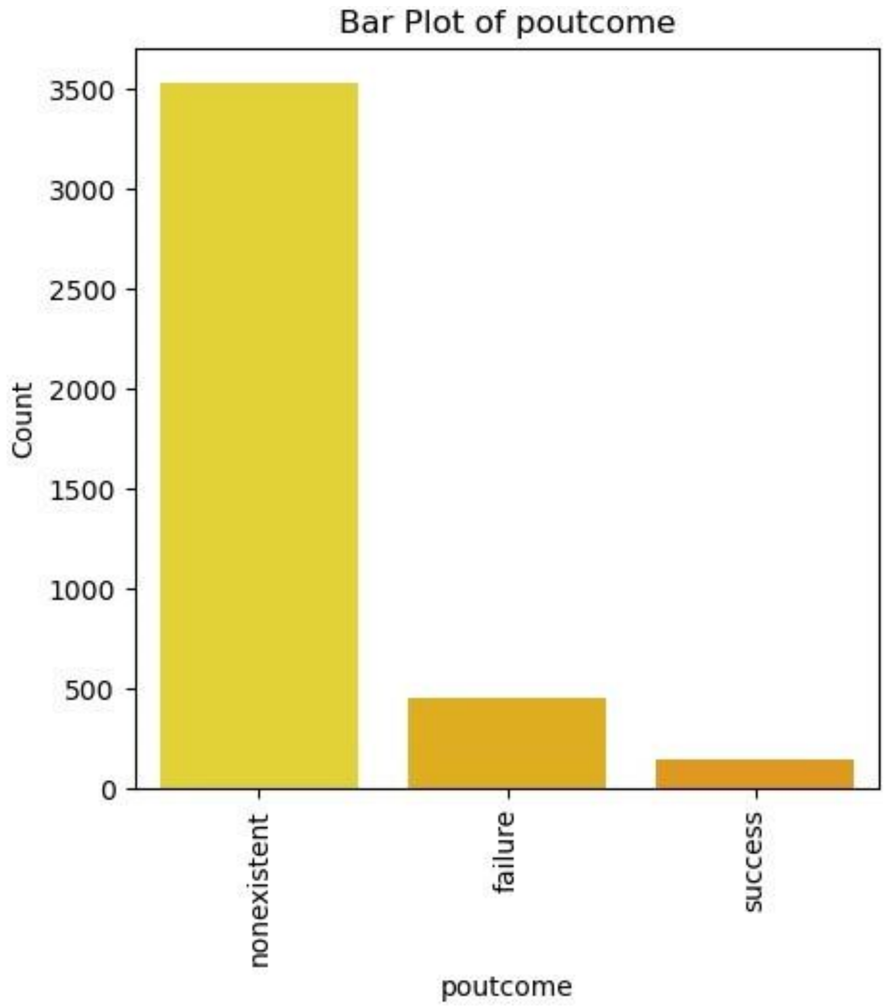

Bar Plot of housing
Count
2000
1500
1000
500
0
yes
no
unknown
housing


Bar Plot of loan
3500
3000
2500
2000
1500
1000
500
0
Count
no
unknown
yes
loan

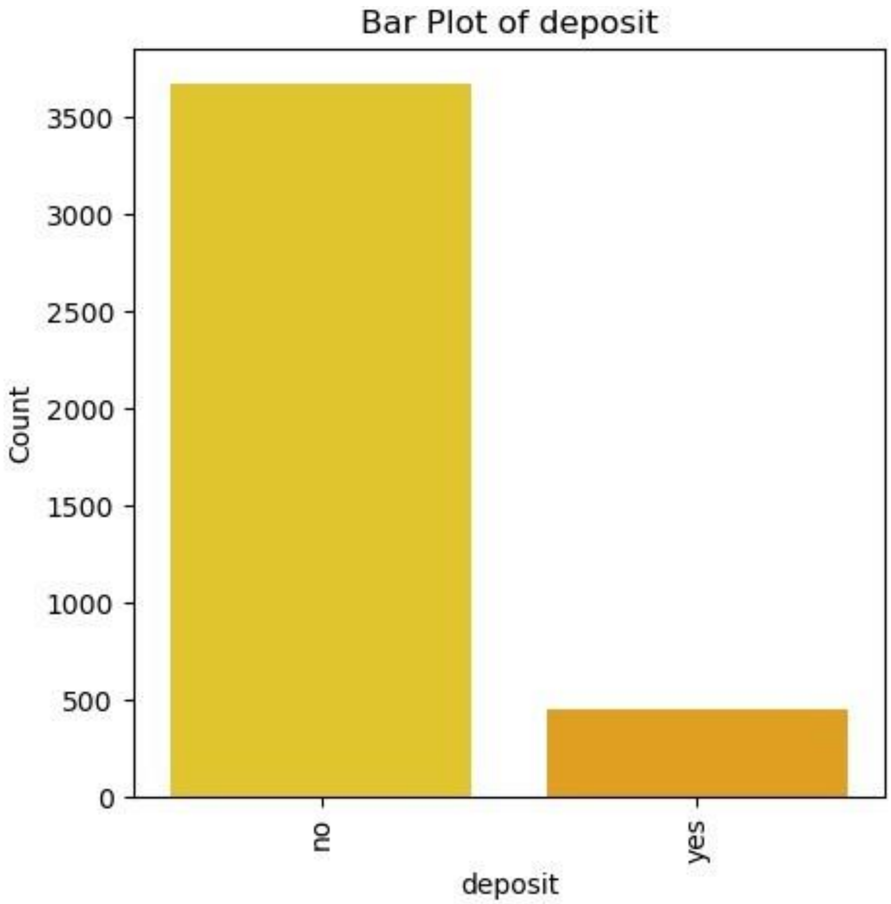

Bar Plot of contact
2500
2000
1500
1000
500
0
Count
cellular
telephone
contact


Bar Plot of month
Count
1400
1200
1000
800
600
400
200
0
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
month


Bar Plot of day_of_week
Count
0
200
400
600
800
fri
wed
mon
thu
tue
day_of_week



Bar Plot of poutcome
3500
3000
2500
2000
1500
1000
500
0
Count
nonexistent
failure
success
poutcome

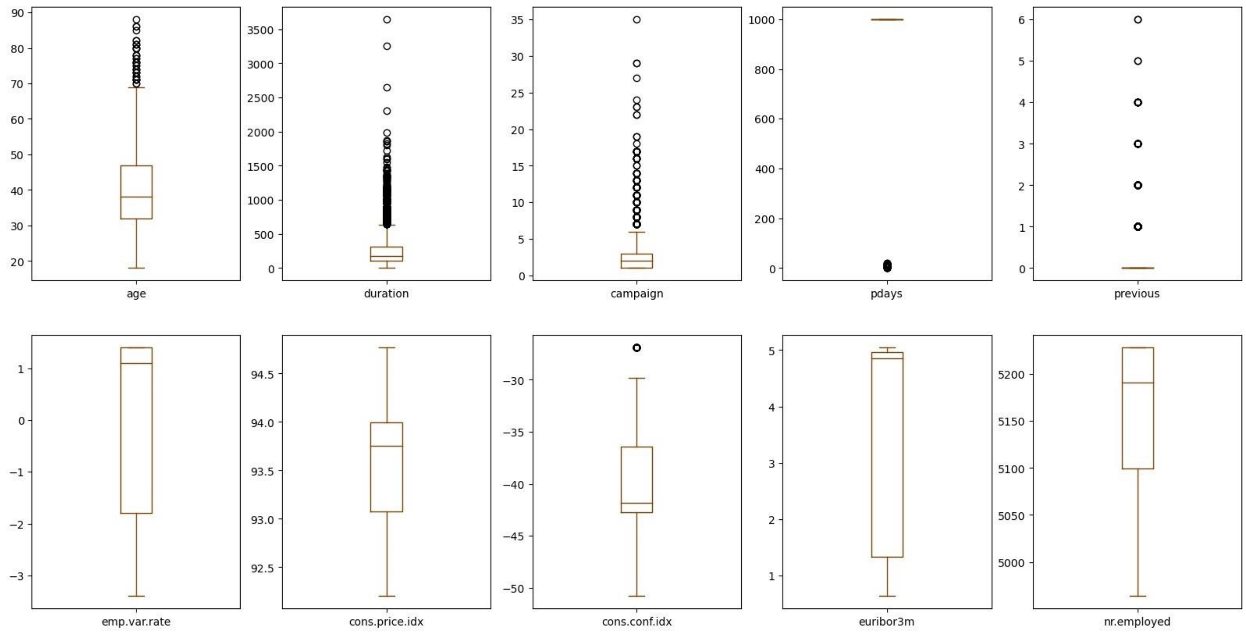


Insights:

1. In the Job Column, we have seen most of the clients are working as 'admin'.
2. In the marital Column, we have seen most of the clients are married.
3. In the education Column, we have seen most of the clients are having 'university.degree' as education.
4. In the default Column, we have seen most of the clients are having 'no' credit as default.
5. In the housing Column, we have seen most of the clients are taking housing loan.
6. In the loan Column, we have seen most of the clients are not taking personal loan.
7. In the contact Column, we have seen most of the clients are choosen cellular as contact.
8. In the month Column, we have seen most of the clients are contacted in the 'may' month.
9. In the day_of_week Column, we have seen most of the clients are contacted in 'thursday'.
10. In the poutcome Column, we have seen the result of most of the previous market campaign is 'nonexistent'.
11. In the target column , we have seen most of the clients are not subscribed a term deposit.

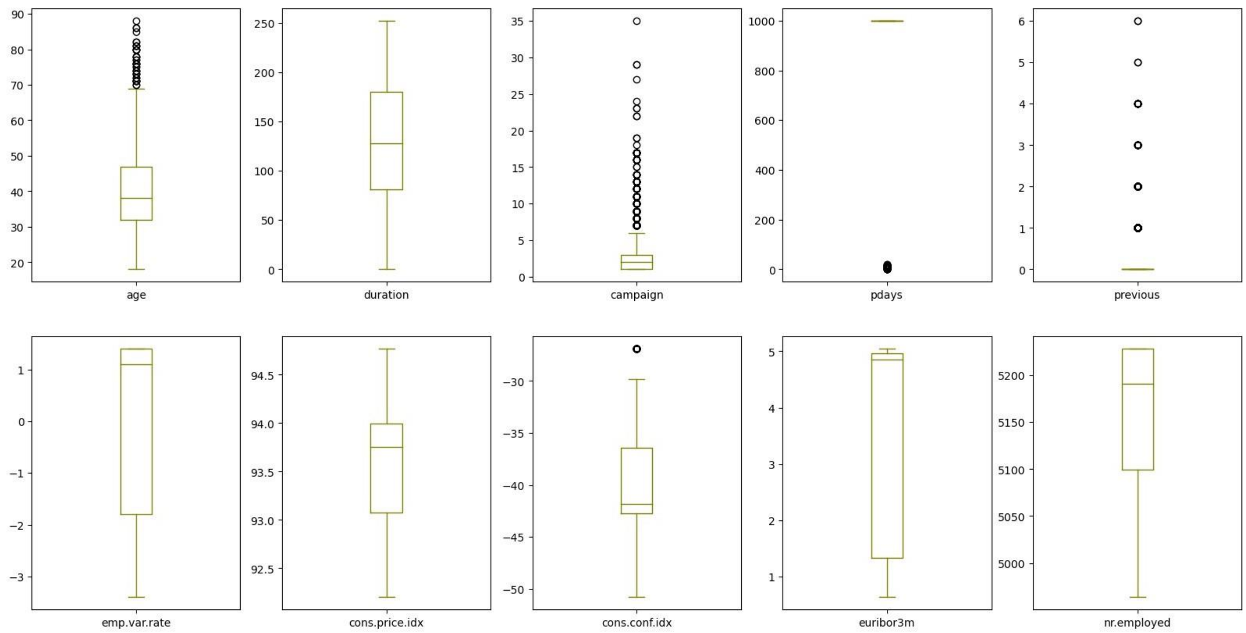
Plotting BoxPlot and Checking for Outliers

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



Through this Plot we can see there are 3 columns having outliers i.e.. 'Age', 'duration' and 'Campaign'. So, we will remove these outliers using Interquantile Range.

```
# Removing outliers

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)]

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



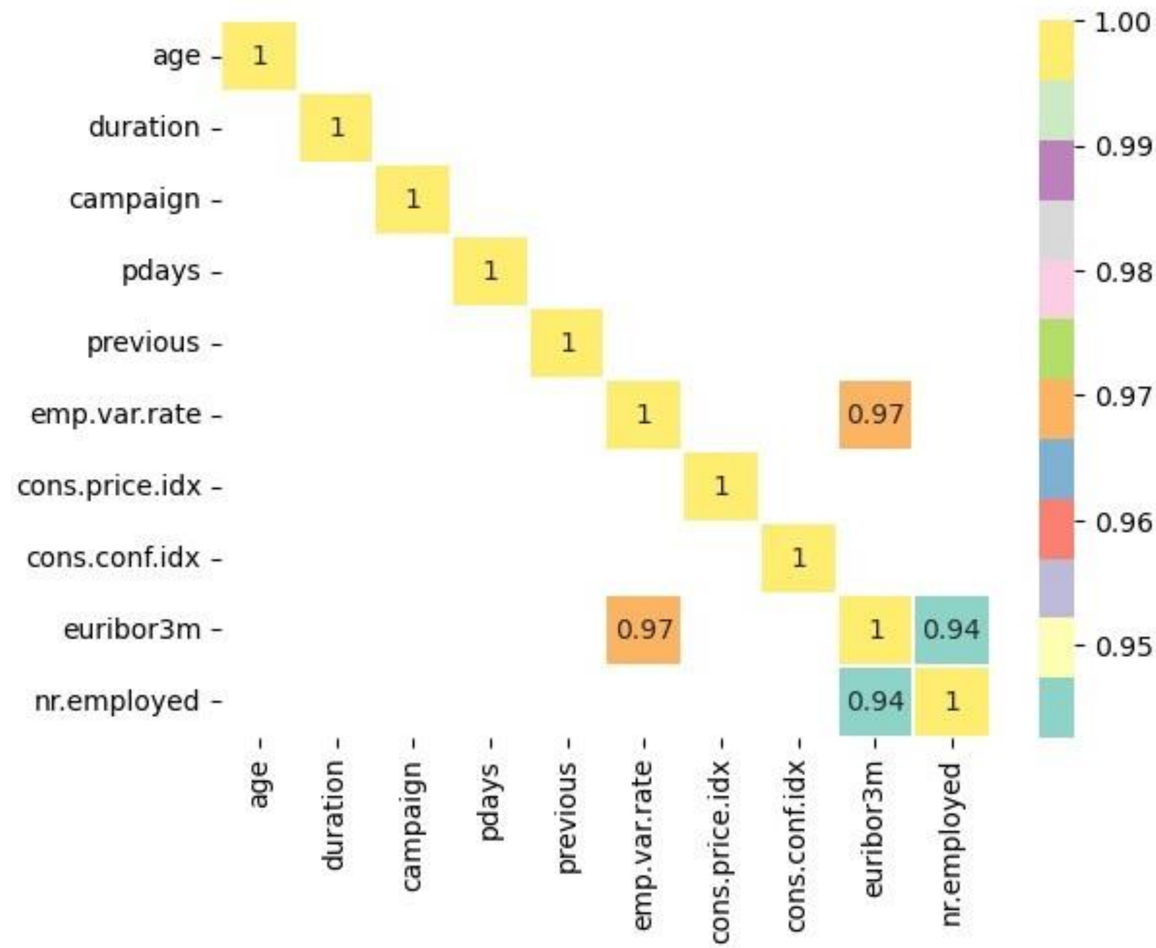
Checking for correlation using Correlation Plot

```
corr = df.corr()
print(corr)
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



Feature Selection using Correlation

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

# copy with original dataframe
df1 = df.copy()
df1.columns

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')

# Removing high correlated columns from the dataset
df1.drop(high_corr_cols,inplace=True,axis=1) # axis=1 indicates
columns
df1.columns
```

```
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')
```

```
# showing dimensions of the updated dataset
df1.shape
```

```
(4119, 18)
```

Now, the dataset is having 4119 rows and 18 columns

Label Encoding

```
# Conversion of categorical columns into numerical columns using label
encoder.
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

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

```
      day_of_week  duration  campaign  pdays  previous  poutcome  \
0               0       250         1     20         0         1
1               0       250         3     20         0         1
2               4       224         0     20         0         1
3               0        14         2     20         0         1
4               1        55         0     20         0         1
...           ...       ...       ...    ...       ...       ...
4114            2        50         0     20         0         1
4115            0       216         0     20         0         1
4116            1        61         1     20         1         0
4117            0       250         0     20         0         1
4118            4       172         0     20         0         1

      cons.price.idx  cons.conf.idx deposit
0                  8              4       0
1                 18             16       0
2                 23              8       0
3                 23              8       0
4                 11              7       0
...              ...            ...     ...
4114              17              6       0
4115              17              6       0
4116               8              4       0
4117              13             17       0
4118              11              7       0

[4119 rows x 18 columns]
```

Checking for target variable

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

```
0    3668
1     451
Name: deposit, dtype: int64
```

Selecting Independent and Dependent Variables

```
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'>
```

Splitting the dataset into Train and Test datasets

```
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)
```

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

Creating a function to compute Confusion Matrix, Classification Report and to generate training and testing scores

```
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))
```

```python
def mscore(model):
    train_score = model.score(x_train,y_train)
    test_score = model.score(x_test,y_test)
    print('Training Score',train_score)  # Training Accuracy
    print('Testing Score',test_score)     # Testing Accuracy
```

Decision Tree Classifier

```python
# Importing  Decision Tree library
from sklearn.tree import DecisionTreeClassifier

# Building Decision Tree Classifier Model
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)
```

```python
# Evaluating training and testing accuracy
mscore(dt)
```

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

```python
# Generating prediction
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

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

```python
# # Evaluate the model - confusion matrix, classification Report,
Accuaracy
eval_model(y_test,ypred_dt)
```

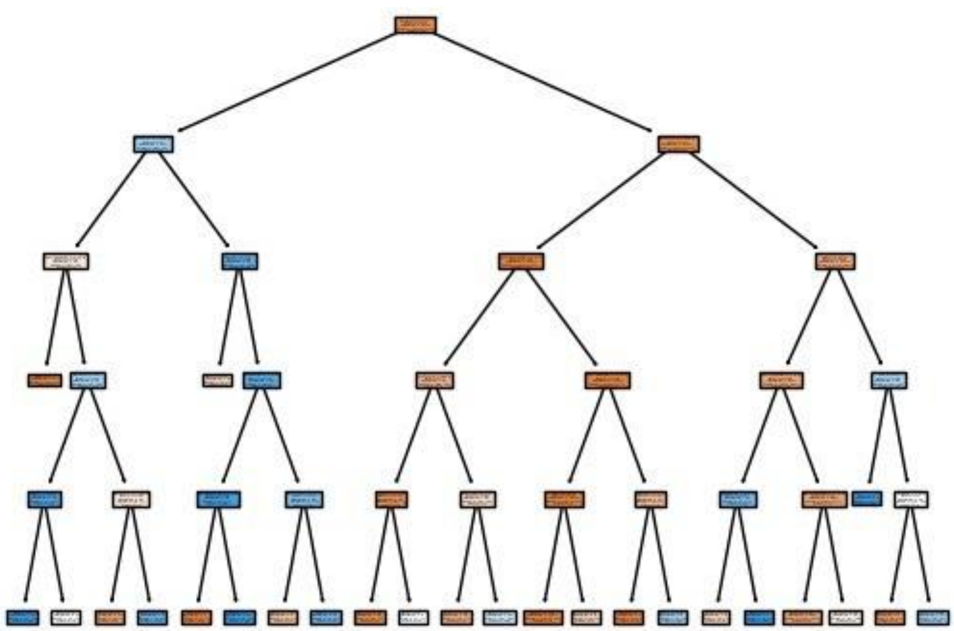
```
Accuracy_Score 0.8990291262135922
Confusion Matrix
 [[905  25]
 [ 79  21]]
Classification Report
               precision    recall  f1-score   support

           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
```

Plot Decision Tree

```
from sklearn.tree import plot_tree

# cn = class names, fn = feature_names
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']
```

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



Decision Tree 2 (using entropy criteria) for visualization

```
# Building Decision Tree Classifier Model
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)
```

```
# Evaluating training and testing accuracy
mscore(dt1)
```

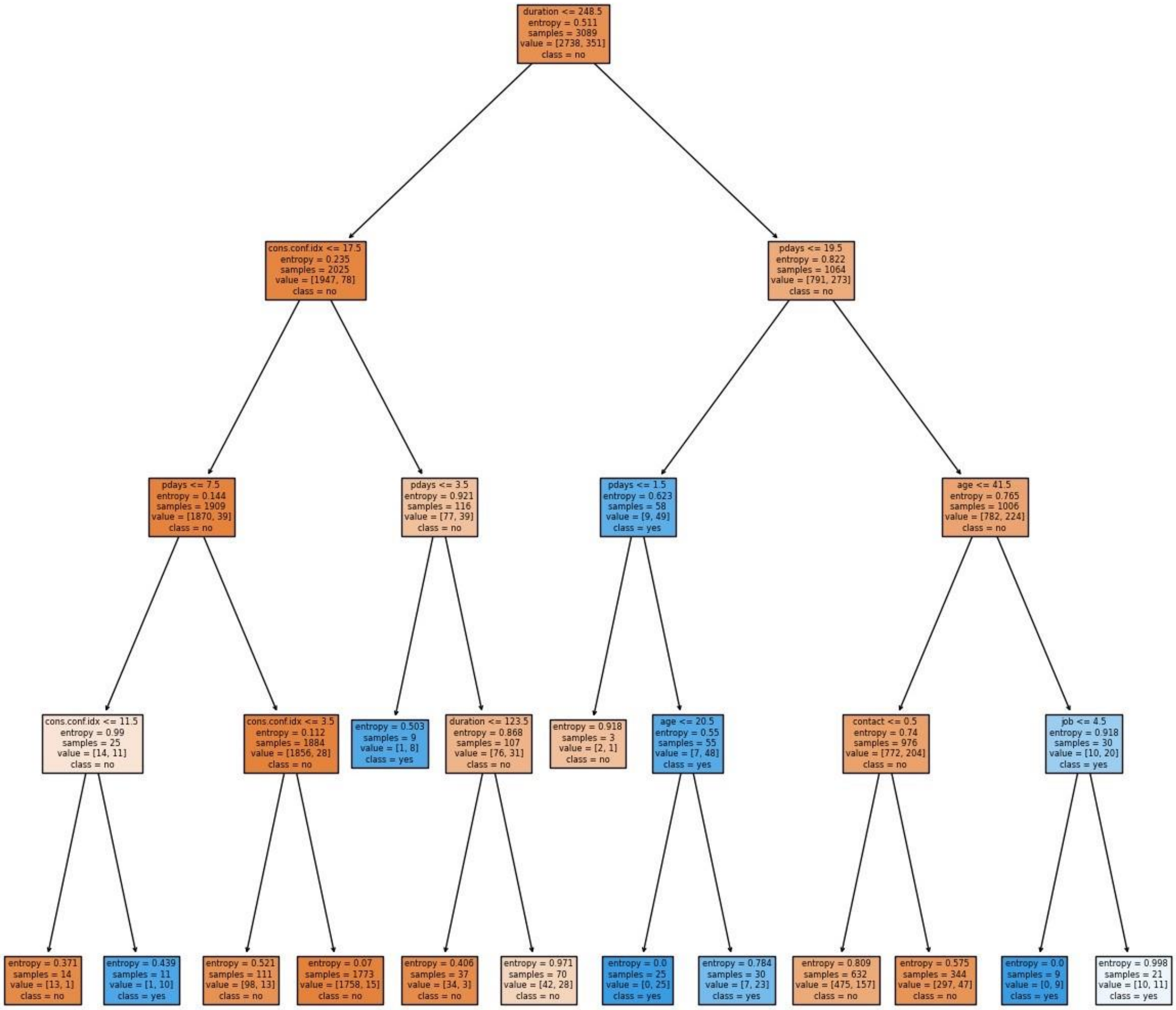
```
Training Score 0.9080608611201036
Testing Score 0.9048543689320389
```

```
# Generating prediction
ypred_dt1 = dt1.predict(x_test)
```

```
# Evaluate the model - confusion matrix, classification Report,
Accuaracy
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
```

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



Conclusion

In this we are having bank marketing dataset, which is sample dataset (10 %) of the original dataset. This dataset contains 4119 rows and 21 columns. In this there is one target variable which is deposit which implies whether the client has taken term deposit or not. The classification goal of this task is to predict if the client will subscribe (yes/no) a term deposit (variable y). In this after performing data cleaning, data preprocessing and feature selection, I have built the decision tree classifier model and evaluated the accuracy of the model which is 90 % using both the criterion (gini) and entropy. This accuracy score is high and the model is good for this dataset.