In [1]:
```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
%matplotlib inline warnings.filterwarnings('ignore')
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
# reading the dataset
df = pd.read_csv("C:\Users\HP\Downloads\bank-additional.csv",delimiter=";")
df.rename(columns={'y':'subscribed_deposit'}, inplace=True)
```

In [3]:
```
df.head()
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	subscribed_deposit
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	0	nonexistent	-1.8	92.893	-46.2	1.313	5099.1	n
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	nonexistent	1.1	93.994	-36.4	4.855	5191.0	n
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4	94.465	-41.8	4.962	5228.1	n
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	0	nonexistent	1.4	94.465	-41.8	4.959	5228.1	n
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	0	nonexistent	-0.1	93.200	-42.0	4.191	5195.8	n

5 rows × 21 columns

In [4]:
```
df.tail()
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	subscribed_deposit
4118	32	admin.	married	high.school	no	yes	yes	cellular	jul	thu	...	1	999	0	nonexistent	1.4	93.918	-42.7	4.968	5228.1	no
4119	39	admin.	married	high.school	no	yes	no	telephone	jul	fri	...	1	999	0	nonexistent	1.4	93.918	-42.7	4.959	5228.1	no
4118	27	student	single	high.school	no	no	no	cellular	may	mon	...	2	999	1	failure	-1.8	92.893	-46.2	1.354	5099.1	no
4117	58	admin.	married	high.school	no	no	no	cellular	aug	fri	...	1	999	0	nonexistent	1.4	93.444	-36.1	4.966	5228.1	no
4118	34	management	single	high.school	no	yes	no	cellular	nov	wed	...	1	999	0	nonexistent	-0.1	93.200	-42.0	4.120	5195.8	no

5 rows × 21 columns

In [5]:
```
# size of the data set
df.shape
```

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

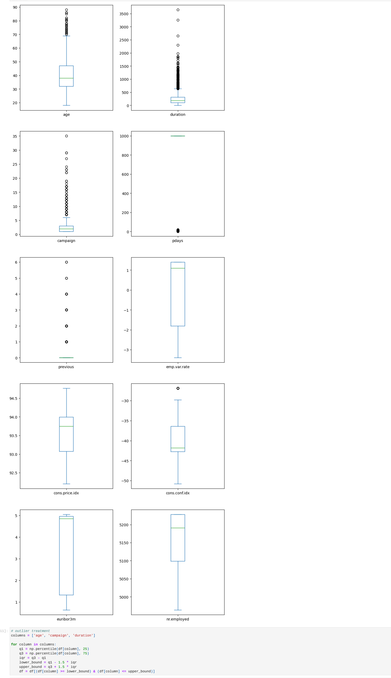
In [6]:
```
# checking for null values
df.isnull().sum()
```

Out[6]:
```
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
subscribed_deposit    0
dtype: int64
```

In [7]:
```
# checking for duplicate values
df.duplicated().sum()
```

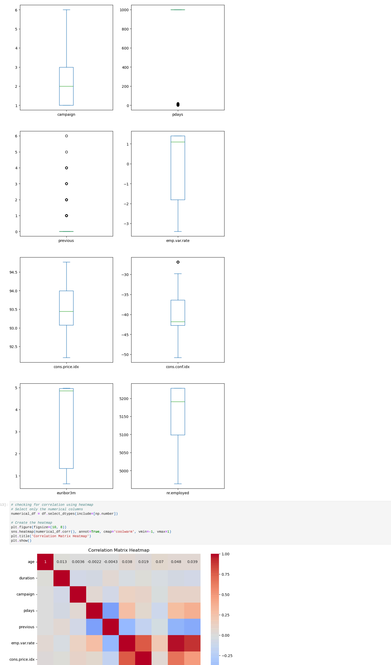
Out[7]:
```
0
```

In [8]:
```
# checking for outliers
df.plot(kind='box', subplots=True, layout=(5,2), figsize=(16,36))
plt.show()
```

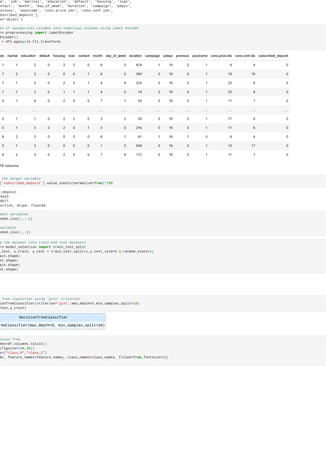


In [9]:
```
# outlier treatment
columns = ['age', 'campaign', 'duration']

for column in columns:
    Q1 = np.percentile(df[column], .25)
    Q3 = np.percentile(df[column], .75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

In [10]:
```
df.plot(kind='box', subplots=True, layout=(5,2), figsize=(16,36))
plt.show()
```



In [11]:
```
# checking for correlation using heatmap
corr = df.corr(numeric_only=True)
numerical_df = df.select_dtypes(include=[np.number])

# Create the heatmap
plt.figure(figsize=(8, 8))
sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



In [12]:
```
high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

In [13]:
```
# copy the original dataframe

df1=df.copy()

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

Out[13]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'cons.price.idx', 'cons.conf.idx',
       'subscribed_deposit'],
      dtype='object')
```

In [14]:
```
# Conversion of categorical columns into numerical columns using label encoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df_encoded = df1.apply(le.fit_transform)
df_encoded
```

Out[14]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	cons.price.idx	cons.conf.idx	subscribed_deposit
0	10	1	1	2	0	2	0	0	6	0	474	1	16	0	1	8	4	0
1	21	7	2	3	0	0	0	1	6	0	363	3	16	0	1	16	16	0
2	7	7	1	3	0	2	0	1	4	4	226	0	16	0	1	20	8	0
3	20	7	1	2	0	1	1	1	4	0	149	2	16	0	1	20	8	0
4	28	0	1	6	0	2	0	0	7	1	55	0	16	0	1	11	7	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4114	12	0	1	1	0	2	2	0	3	2	50	0	16	0	1	17	6	0
4115	21	0	1	3	0	2	0	1	3	0	216	0	16	0	1	17	6	0
4116	8	0	2	3	0	0	0	0	6	1	61	0	16	1	0	8	4	0
4117	42	0	1	3	0	0	0	0	1	0	509	0	16	0	1	13	17	0
4118	16	0	2	3	0	2	0	0	7	4	172	0	16	0	1	11	7	0

3272 rows × 18 columns

In [15]:
```
# Checking the target variable
df_encoded['subscribed_deposit'].value_counts(normalize=True)*100
```

Out[15]:
```
subscribed_deposit
0    92.218215
1     7.781785
Name: proportions, dtype: float64
```

In [16]:
```
# independent variables
x = df_encoded.iloc[:,:-1]

# target variable
y = df_encoded.iloc[:,-1]
```

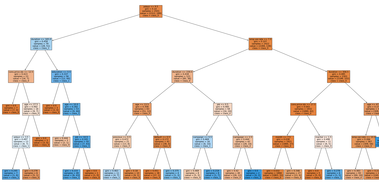
In [17]:
```
# Splitting the dataset into train and test datasets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state=1)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2291, 17)
(1277, 17)
(2291,)
(1277,)
```

In [18]:
```
# decision tree classifier using 'gini' criterion
dc = DecisionTreeClassifier(criterion='gini',max_depth=5,min_samples_split=10)
dc.fit(x_train,y_train)
```

Out[18]:
```
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

In [19]:
```
# Plot Decision Tree
plt.figure(figsize=(20,10),dpi=300)
plot_tree(dc, feature_names=feature_names, class_names=class_names, filled=True, fontsize=12)
plt.show()
```



In [ ]: