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 [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

New Section

In [3]:
```
df = pd.read_csv('bank.csv', sep=';')
df.head()
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	mor
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	m
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	a
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	.
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	m

EDA

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        4521 non-null   int64 
 1   job        4521 non-null   object
 2   marital    4521 non-null   object
 3   education  4521 non-null   object
 4   default    4521 non-null   object
 5   balance    4521 non-null   int64 
 6   housing    4521 non-null   object
 7   loan       4521 non-null   object
 8   contact    4521 non-null   object
 9   day        4521 non-null   int64 
 10  month      4521 non-null   object
 11  duration   4521 non-null   int64 
 12  campaign   4521 non-null   int64 
 13  pdays      4521 non-null   int64 
 14  previous   4521 non-null   int64 
 15  poutcome   4521 non-null   object
 16  y          4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
```

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

Out[5]:

	age	job	marital	education	default	balance	housing	loan	contact	day
4516	33	services	married	secondary	no	-333	yes	no	cellular	30
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9
4518	57	technician	married	secondary	no	295	no	no	cellular	19
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3

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

Out[6]:
```
age            67
job            12
marital         3
education       4
default         2
balance      2353
housing         2
loan            2
contact         3
day            31
month          12
duration      875
campaign       32
pdays         292
previous       24
poutcome        4
y               2
dtype: int64
```

In [7]:
```
df.describe()
```

Out[7]:

	age	balance	day	duration	campaign	pdays	p
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.

In [8]:
```
# chicking for duplicites
df[df.duplicated()].count()
```

Out[8]:
```
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
y            0
dtype: int64
```

In [9]:
```
#check for missing values
df.isnull().sum()
```

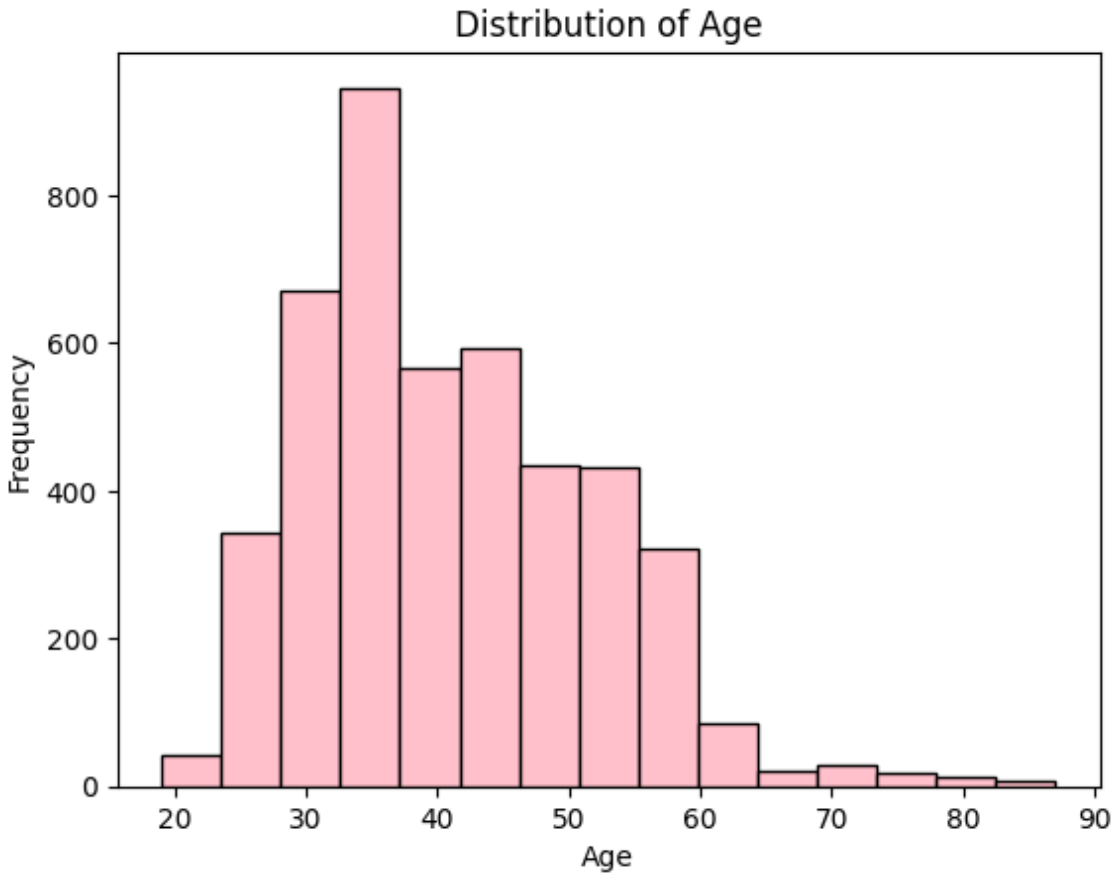
Out[9]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [10]:
```
# dropping unnecissery columns
df.drop(['day', 'month'], axis=1, inplace=True)
df.head()
```

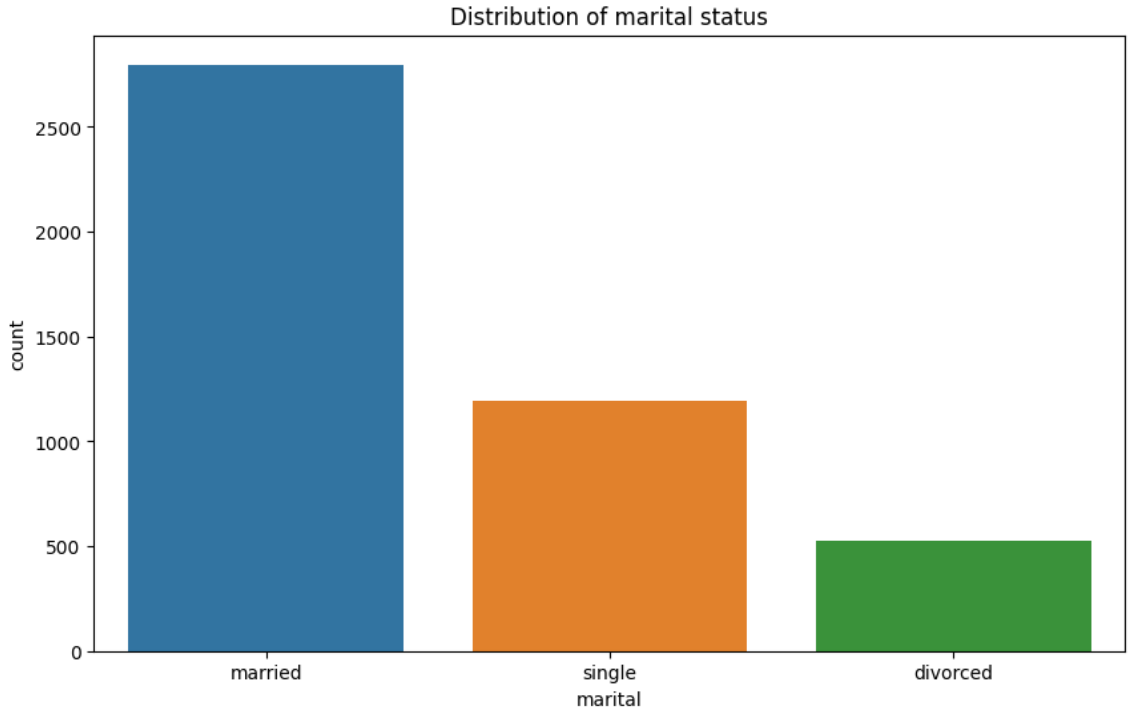
Out[10]:

	age	job	marital	education	default	balance	housing	loan	contact	duration
0	30	unemployed	married	primary	no	1787	no	no	cellular	79
1	33	services	married	secondary	no	4789	yes	yes	cellular	220
2	35	management	single	tertiary	no	1350	yes	no	cellular	185
3	30	management	married	tertiary	no	1476	yes	yes	unknown	199
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	226

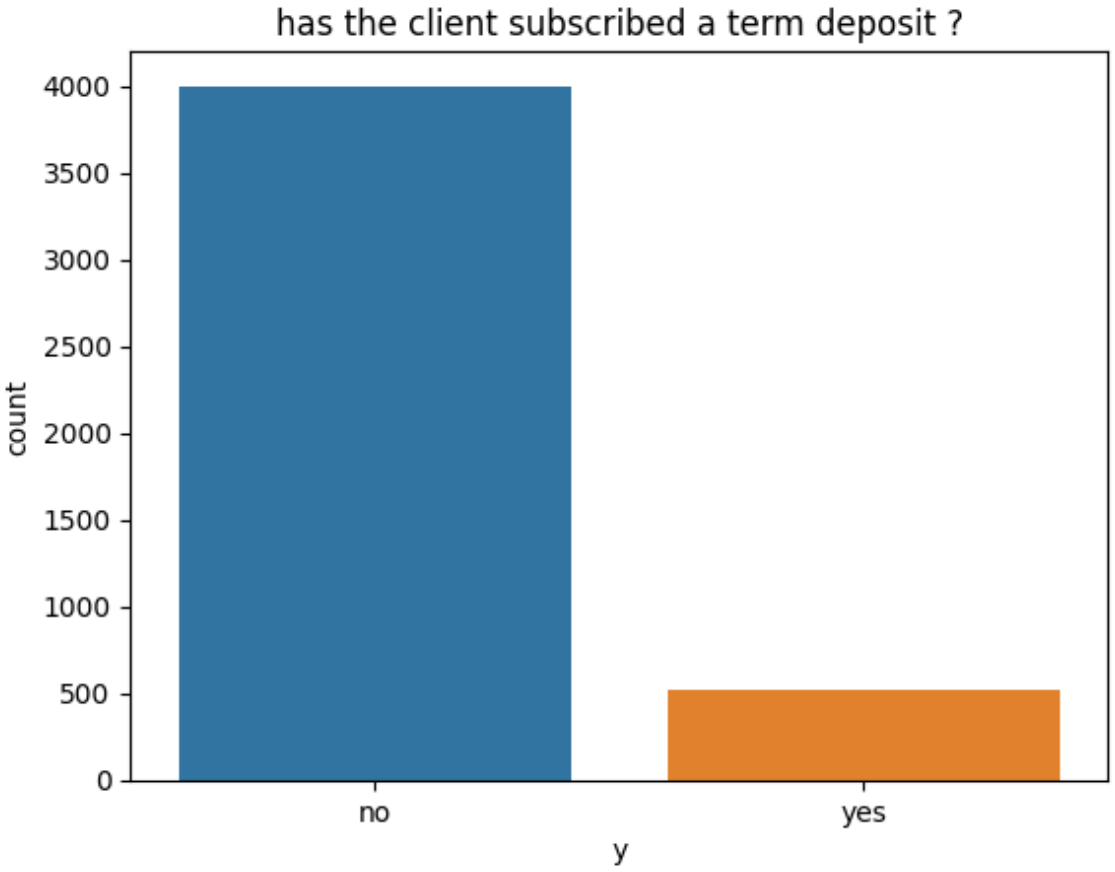
In [12]:
```
# distribution of Age
plt.figure()
plt.hist(df['age'], bins=15, color='pink', edgecolor='black')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



In [13]:
```
# relationships
plt.figure(figsize=(10, 6))
sns.countplot(x='marital', data=df)
plt.title('Distribution of marital status')
plt.show()
```



In [14]:
```
# target vlaue
plt.title('has the client subscribed a term deposit ?')
sns.countplot(x='y', data=df)
plt.show()
```



In [15]:
```
X = df.drop('y', axis=1)
y = df['y']
X = pd.get_dummies(X, columns=['job', 'marital', 'education', 'default', 'h
y_binary = y.map({'yes': 1, 'no': 0})
```

In [16]:
```
# decising tree
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=

dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)
y_pred_dt = dt_classifier.predict(X_test)

# Store evaluation metrics
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(accuracy_dt)
```

```
0.850828729281768
```

In [ ]: