In [ ]:

```python
# Bank Marketing Data Science & Machine Learning Project
```

In [13]:

```python
## Step 1: Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
```

In [14]:

```python
## Step 2: Load Dataset
df = pd.read_csv('bank-additional-full.csv', sep=';')
print(df.shape)
print(df.head())
print(df.info())
print(df['y'].value_counts())
```

```
(41188, 21)
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
3   40     admin.  married     basic.6y       no      no   no  telephone
4   56   services  married  high.school       no      no  yes  telephone

  month day_of_week  ...  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  ...         1    999         0  nonexistent          1.1
1   may         mon  ...         1    999         0  nonexistent          1.1
2   may         mon  ...         1    999         0  nonexistent          1.1
3   may         mon  ...         1    999         0  nonexistent          1.1
4   may         mon  ...         1    999         0  nonexistent          1.1

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
None
y
no     36548
yes     4640
Name: count, dtype: int64
```

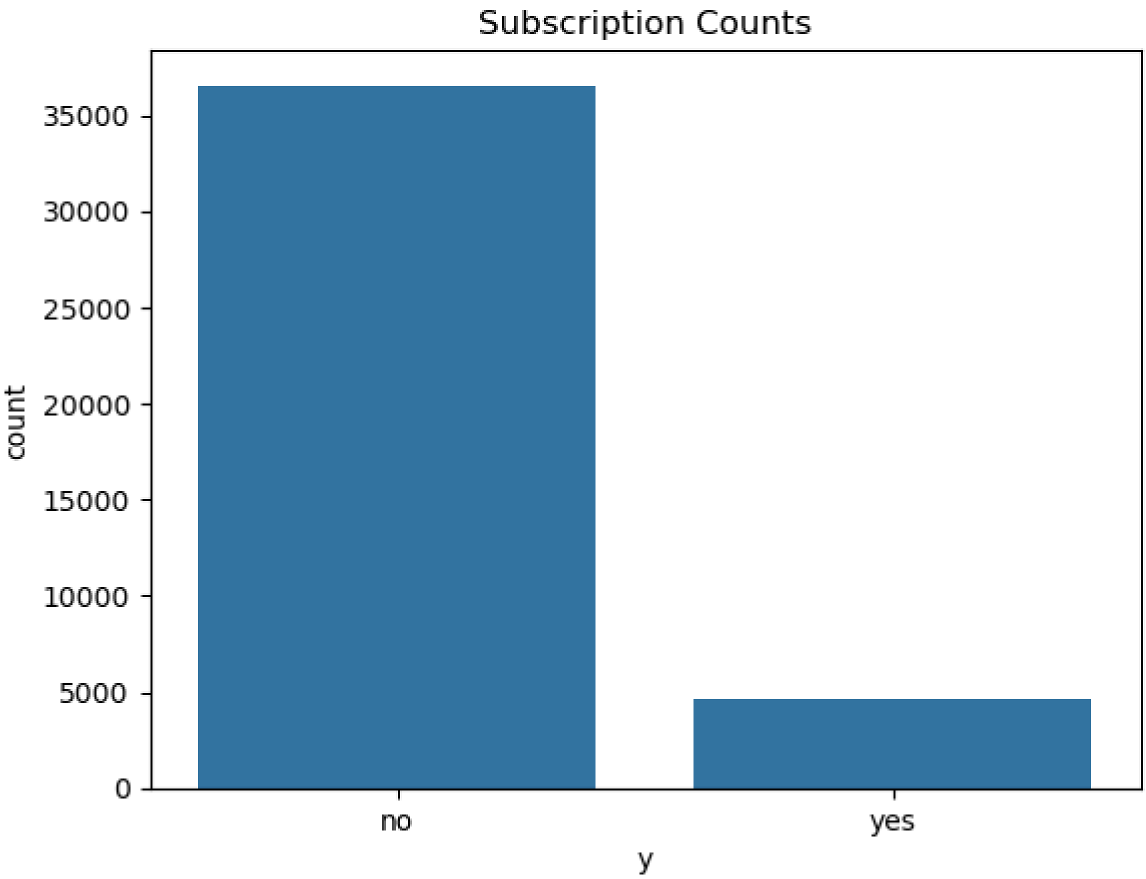
In [15]:

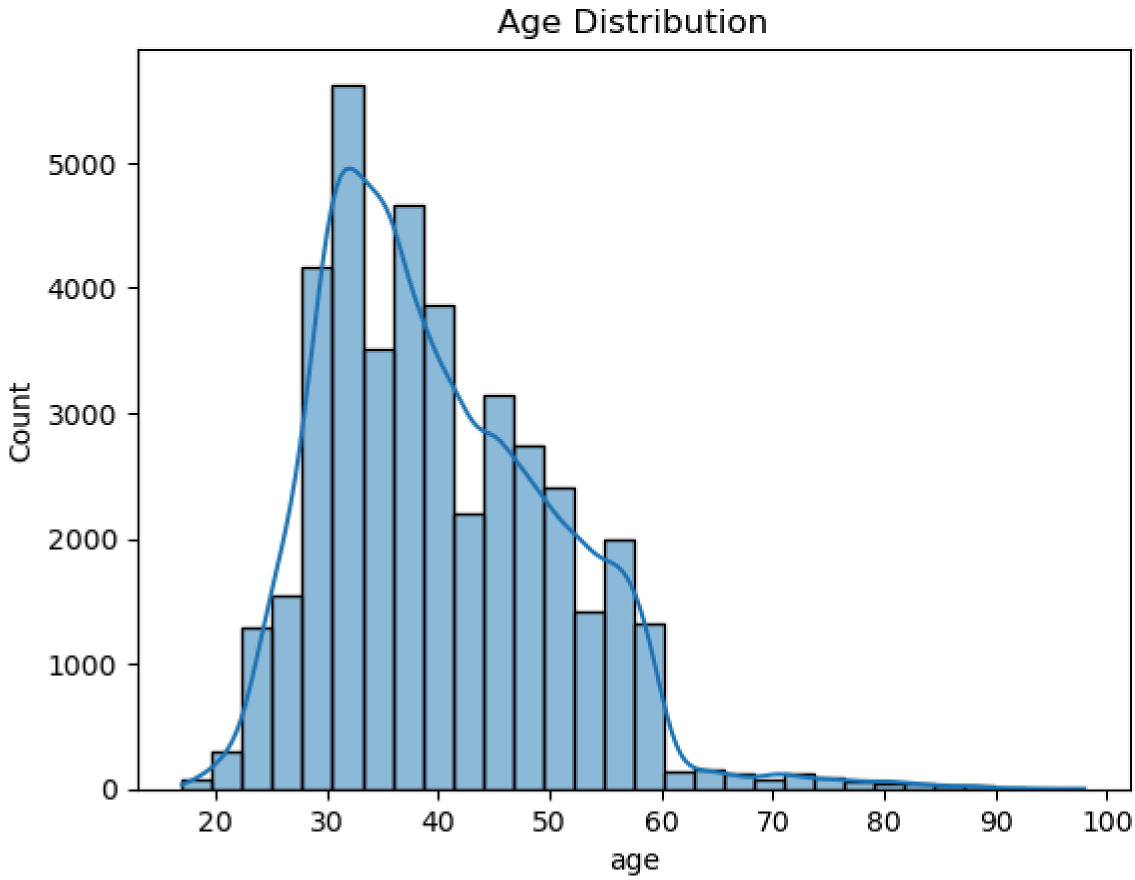
```
## Step 3: Exploratory Data Analysis (EDA)
sns.countplot(x='y', data=df)
plt.title('Subscription Counts')
```

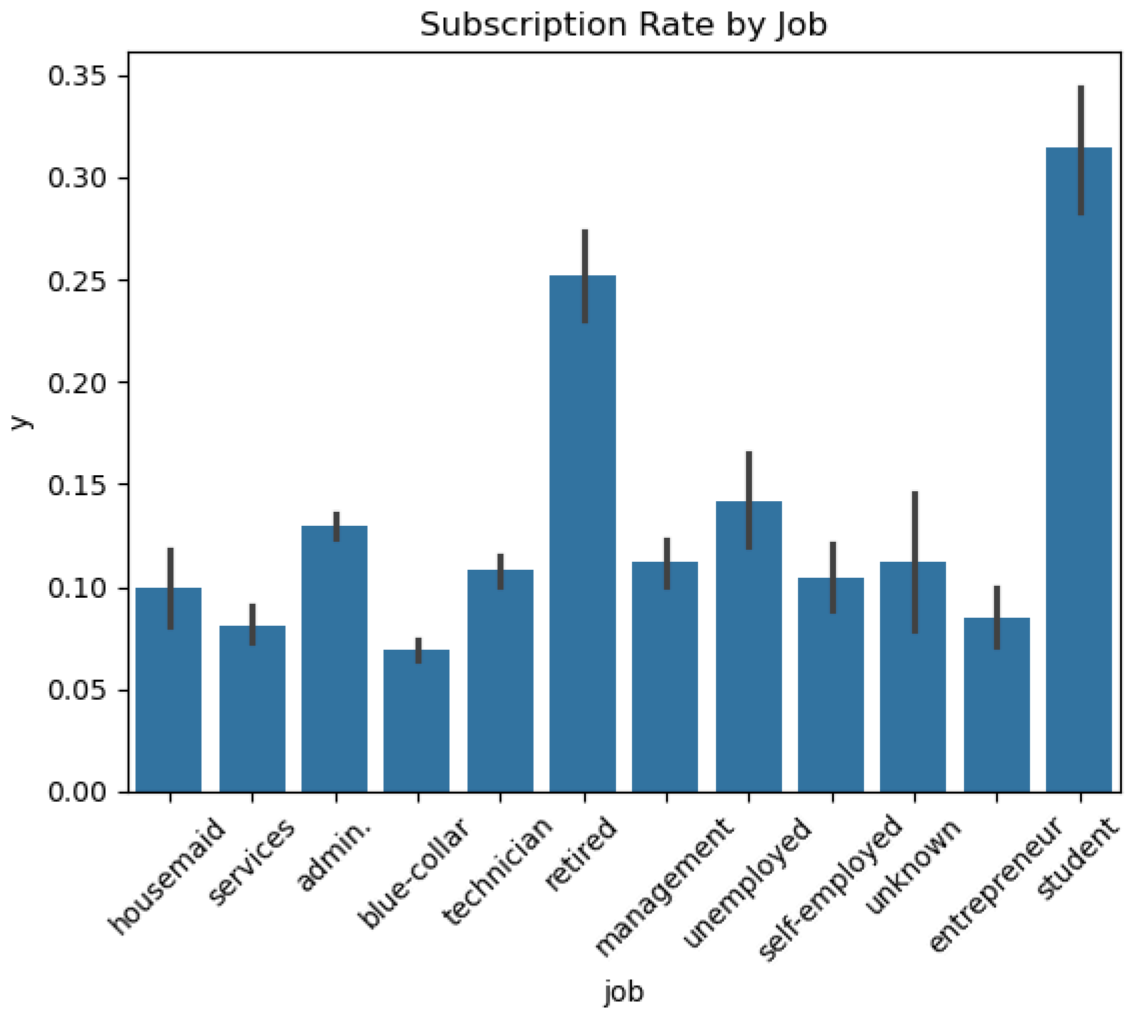
```
plt.show()

sns.histplot(df['age'], bins=30, kde=True)
plt.title('Age Distribution')
plt.show()

sns.barplot(x='job', y=df['y'].apply(lambda x: 1 if x == 'yes' else 0), data=df)
plt.xticks(rotation=45)
plt.title('Subscription Rate by Job')
plt.show()
```




Age Distribution
Count
0
1000
2000
3000
4000
5000
20
30
40
50
60
70
80
90
100
age


Subscription Rate by Job
y
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
housemaid
services
admin.
blue-collar
technician
retired
management
unemployed
self-employed
unknown
entrepreneur
student
job

In [19]:
```
## Step 4: Data Preprocessing
# Encode target
le = LabelEncoder()
df['y'] = le.fit_transform(df['y'])  # yes=1, no=0
```

In [20]:
```
# Encode categorical features
df_encoded = pd.get_dummies(df.drop('y', axis=1))
X = df_encoded
y = df['y']
```

In [21]:
```
## Step 5: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [22]:
```
## Step 6: Train Random Forest
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

Out[22]:
```
▾ RandomForestClassifier

RandomForestClassifier(random_state=42)
```

In [23]:
```
## Step 7: Evaluate Model
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("ROC AUC:", roc_auc_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      7303
           1       0.65      0.45      0.53       935

    accuracy                           0.91      8238
   macro avg       0.79      0.71      0.74      8238
weighted avg       0.90      0.91      0.90      8238

ROC AUC: 0.7087280079024003
```

In [26]:
```
## Step 8: Feature Importance
importances = model.feature_importances_
features = X.columns
sorted_indices = importances.argsort()[::-1]

print("Top Features:")
for i in sorted_indices[:10]:
    print(f"{features[i]}: {importances[i]:.4f}")
```

```
Top Features:
duration: 0.2759
euribor3m: 0.0957
age: 0.0790
nr.employed: 0.0481
campaign: 0.0395
pdays: 0.0323
emp.var.rate: 0.0259
cons.conf.idx: 0.0253
cons.price.idx: 0.0224
poutcome_success: 0.0158
```

In [ ]:

```

```