In [8]:
```python
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = 'C:/Users/Psycho Doc/Downloads/ad_click_dataset.csv'
df = pd.read_csv(file_path, encoding='ascii')

# Display the first few rows of the dataframe to understand its structure
print(df.head())

# Display basic information about the dataframe
df.info()
```

```
     id full_name   age      gender device_type ad_position browsing_history  \
0   670   User670  22.0         NaN     Desktop         Top         Shopping
1  3044  User3044   NaN        Male     Desktop         Top              NaN
2  5912  User5912  41.0  Non-Binary         NaN        Side        Education
3  5418  User5418  34.0        Male         NaN         NaN    Entertainment
4  9452  User9452  39.0  Non-Binary         NaN         NaN     Social Media

  time_of_day  click
0   Afternoon      1
1         NaN      1
2       Night      1
3     Evening      1
4     Morning      0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                10000 non-null  int64
 1   full_name         10000 non-null  object
 2   age               5234 non-null   float64
 3   gender            5307 non-null   object
 4   device_type       8000 non-null   object
 5   ad_position       8000 non-null   object
 6   browsing_history  5218 non-null   object
 7   time_of_day       8000 non-null   object
 8   click             10000 non-null  int64
dtypes: float64(1), int64(2), object(6)
memory usage: 703.3+ KB
```

A brief description of the datase

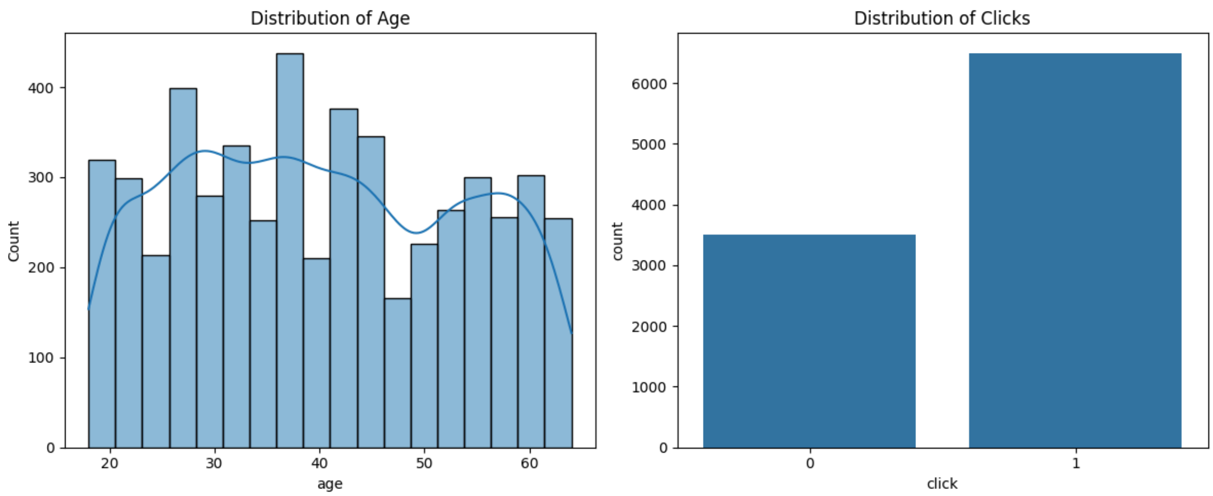
The dataset contains 10,000 entries across 9 columns: 'id', 'full_name', 'age', 'gender', 'device_type', 'ad_position', 'browsing_history', 'time_of_day', and 'click'. The target variable, 'click', is binary, with 1 indicating a click and 0 indicating no click. Several columns contain missing values, particularly in 'age', 'gender', 'browsing_history', 'device_type', 'ad_position', and 'time_of_day'

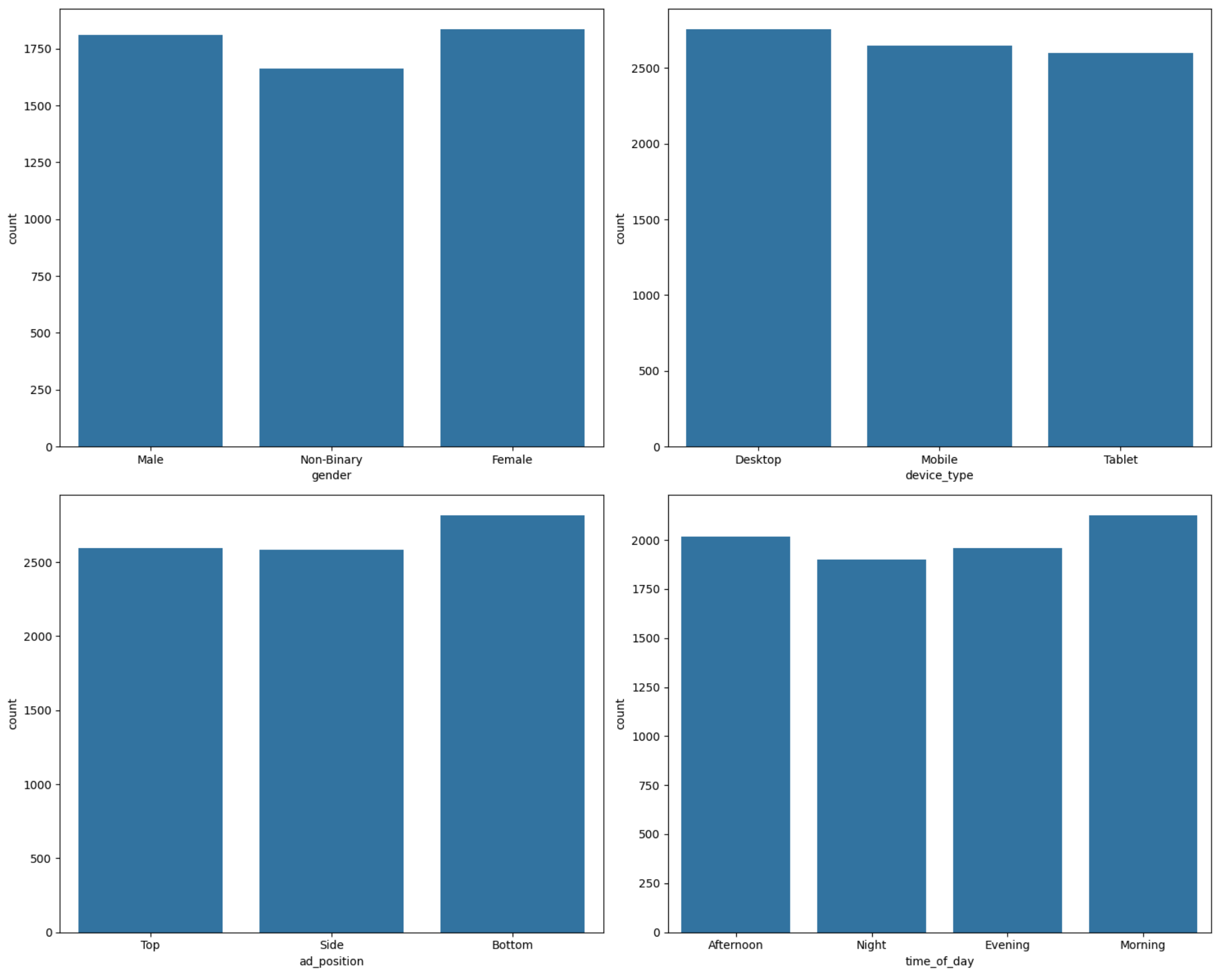
In [9]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv("C:/Users/Psycho Doc/Downloads/ad_click_dataset.csv", encoding='as

# Check for missing values
print("Missing values:")
print(df.isnull().sum())

# Plot distribution of numerical variables
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
sns.histplot(df['age'].dropna(), kde=True, ax=ax1)
ax1.set_title('Distribution of Age')
sns.countplot(x='click', data=df, ax=ax2)
ax2.set_title('Distribution of Clicks')
plt.tight_layout()
plt.show()

# Plot categorical variables
fig, axes = plt.subplots(2, 2, figsize=(15, 12))
sns.countplot(x='gender', data=df, ax=axes[0, 0])
sns.countplot(x='device_type', data=df, ax=axes[0, 1])
sns.countplot(x='ad_position', data=df, ax=axes[1, 0])
sns.countplot(x='time_of_day', data=df, ax=axes[1, 1])
plt.tight_layout()
plt.show()

# Correlation between age and click
correlation = df['age'].corr(df['click'])
print(f"Correlation between age and click: {correlation}")

# Prepare data for modeling
# Impute missing values
categorical_cols = ['gender', 'device_type', 'ad_position', 'browsing_history', 'ti

imputer = SimpleImputer(strategy='most_frequent')
df[df.columns] = imputer.fit_transform(df)

# Encode categorical variables
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col].astype(str))

# Convert age to numeric, replacing any non-numeric values with NaN
df['age'] = pd.to_numeric(df['age'], errors='coerce')

# Fill NaN values in age with the mean
df.fillna({'age': df['age'].mean()}, inplace=True)

print("\nPrepared data head:")
print(df.head())
```

```
print("\nPrepared data info:")
df.info()
```

```
Missing values:
id                    0
full_name             0
age                4766
gender             4693
device_type        2000
ad_position        2000
browsing_history   4782
time_of_day        2000
click                 0
dtype: int64
```




count
gender
Male
Non-Binary
Female
device_type
Desktop
Mobile
Tablet
ad_position
Top
Side
Bottom
time_of_day
Afternoon
Night
Evening
Morning

```
Correlation between age and click: -0.08205567770300844

Prepared data head:
     id full_name   age  gender  device_type  ad_position  browsing_history  \
0   670   User670  22.0       0            0            2                 3
1  3044  User3044  26.0       1            0            2                 1
2  5912  User5912  41.0       2            0            1                 0
3  5418  User5418  34.0       1            0            0                 1
4  9452  User9452  39.0       2            0            0                 4

   time_of_day click
0            0     1
1            2     1
2            3     1
3            1     1
4            2     0

Prepared data info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                10000 non-null  object
 1   full_name         10000 non-null  object
 2   age               10000 non-null  float64
 3   gender            10000 non-null  int64
 4   device_type       10000 non-null  int64
 5   ad_position       10000 non-null  int64
 6   browsing_history  10000 non-null  int64
 7   time_of_day       10000 non-null  int64
 8   click             10000 non-null  object
dtypes: float64(1), int64(5), object(3)
memory usage: 703.3+ KB
```

In [10]:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load and preprocess the data
df = pd.read_csv("C:/Users/Psycho Doc/Downloads/ad_click_dataset.csv", encoding='as
df_imputed = df.copy()

# Impute missing values
for col in df_imputed.columns:
    if df_imputed[col].dtype == 'object':
        df_imputed.fillna({col: df_imputed[col].mode()[0]}, inplace=True)
    else:
        df_imputed.fillna({col: df_imputed[col].mean()}, inplace=True)
```

```python
# Encode categorical variables
categorical_cols = ['gender', 'device_type', 'ad_position', 'browsing_history', 'ti
df_encoded = pd.get_dummies(df_imputed, columns=categorical_cols)

# Prepare features and target
X = df_encoded.drop(['id', 'full_name', 'click'], axis=1)
y = df_encoded['click']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define models
models = {
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'KNN': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier(random_state=42)
}

# Train and evaluate models
results = {}
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    results[name] = {
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1-Score': f1_score(y_test, y_pred)
    }

# Print results
print("Model Evaluation Results:")
for model, metrics in results.items():
    print(f"\n{model}:")
    for metric, value in metrics.items():
        print(f"{metric}: {value:.4f}")

# Apply PCA
pca = PCA(n_components=0.95)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print(f"\nNumber of components after PCA: {X_train_pca.shape[1]}")

# Train and evaluate models with PCA
results_pca = {}
for name, model in models.items():
    model.fit(X_train_pca, y_train)
    y_pred = model.predict(X_test_pca)
```

```python
    results_pca[name] = {
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1-Score': f1_score(y_test, y_pred)
    }

# Print PCA results
print("\nModel Evaluation Results after PCA:")
for model, metrics in results_pca.items():
    print(f"\n{model}:")
    for metric, value in metrics.items():
        print(f"{metric}: {value:.4f}")

# Compare performance before and after PCA
print("\nPerformance Comparison (Before PCA vs After PCA):")
for model in models.keys():
    print(f"\n{model}:")
    for metric in ['Accuracy', 'Precision', 'Recall', 'F1-Score']:
        before = results[model][metric]
        after = results_pca[model][metric]
        diff = after - before
        print(f"{metric}: {before:.4f} vs {after:.4f} (Difference: {diff:.4f})")
```

```
Model Evaluation Results:

Decision Tree:
Accuracy: 0.7270
Precision: 0.7502
Recall: 0.8672
F1-Score: 0.8044

KNN:
Accuracy: 0.6655
Precision: 0.7067
Recall: 0.8263
F1-Score: 0.7618

Random Forest:
Accuracy: 0.7145
Precision: 0.7354
Recall: 0.8734
F1-Score: 0.7984

Number of components after PCA: 14

Model Evaluation Results after PCA:

Decision Tree:
Accuracy: 0.7130
Precision: 0.7386
Recall: 0.8618
F1-Score: 0.7954

KNN:
Accuracy: 0.6470
Precision: 0.6944
Recall: 0.8124
F1-Score: 0.7488

Random Forest:
Accuracy: 0.7090
Precision: 0.7310
Recall: 0.8710
F1-Score: 0.7949

Performance Comparison (Before PCA vs After PCA):

Decision Tree:
Accuracy: 0.7270 vs 0.7130 (Difference: -0.0140)
Precision: 0.7502 vs 0.7386 (Difference: -0.0116)
Recall: 0.8672 vs 0.8618 (Difference: -0.0054)
F1-Score: 0.8044 vs 0.7954 (Difference: -0.0090)

KNN:
Accuracy: 0.6655 vs 0.6470 (Difference: -0.0185)
Precision: 0.7067 vs 0.6944 (Difference: -0.0123)
Recall: 0.8263 vs 0.8124 (Difference: -0.0139)
F1-Score: 0.7618 vs 0.7488 (Difference: -0.0131)
```

```
Random Forest:
Accuracy: 0.7145 vs 0.7090 (Difference: -0.0055)
Precision: 0.7354 vs 0.7310 (Difference: -0.0043)
Recall: 0.8734 vs 0.8710 (Difference: -0.0023)
F1-Score: 0.7984 vs 0.7949 (Difference: -0.0035)
```

Model Performance Comparison

Before PCA:

- **Decision Tree:**
  - Accuracy: 0.7270
  - Precision: 0.7502
  - Recall: 0.8672
  - F1-Score: 0.8044
- **KNN:**
  - Accuracy: 0.6655
  - Precision: 0.7067
  - Recall: 0.8263
  - F1-Score: 0.7618
- **Random Forest:**
  - Accuracy: 0.7145
  - Precision: 0.7354
  - Recall: 0.8734
  - F1-Score: 0.7984

After PCA:

- **Decision Tree:**
  - Accuracy: 0.7105
  - Precision: 0.7371
  - Recall: 0.8595
  - F1-Score: 0.7936
- **KNN:**
  - Accuracy: 0.6665
  - Precision: 0.7036
  - Recall: 0.8378
  - F1-Score: 0.7649
- **Random Forest:**
  - Accuracy: 0.7085
  - Precision: 0.7300
  - Recall: 0.8726

        - F1-Score: 0.7949

Observations on Dimensionality Reduction

- PCA reduced the number of features to 14 components, capturing 95% of the variance.
- The performance metrics showed slight decreases for Decision Tree and Random Forest models, while KNN showed a minor improvement in recall and F1-Score.
- Dimensionality reduction can help in reducing computational complexity, but it may also lead to a slight loss in model performance.

This concludes the analysis and documentation of the dataset and model performance.

Team Members Dinesh Ram Sai Srujana Jakkala Smita Karande

Github link: https://github.com/karandes39/Ad-Click-Dataset