Decision tree classifier

Import Libraries

In [33]:

```
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 classification_report, confusion_matrix, accurac
from sklearn.tree import plot_tree
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```

Load and preprocess the dataset

In [3]:

```
# Load dataset
df = pd.read_csv('bank.csv')
df.head()
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        11162 non-null  int64 
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64 
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64 
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64 
 12  campaign   11162 non-null  int64 
 13  pdays      11162 non-null  int64 
 14  previous   11162 non-null  int64 
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

In [17]:

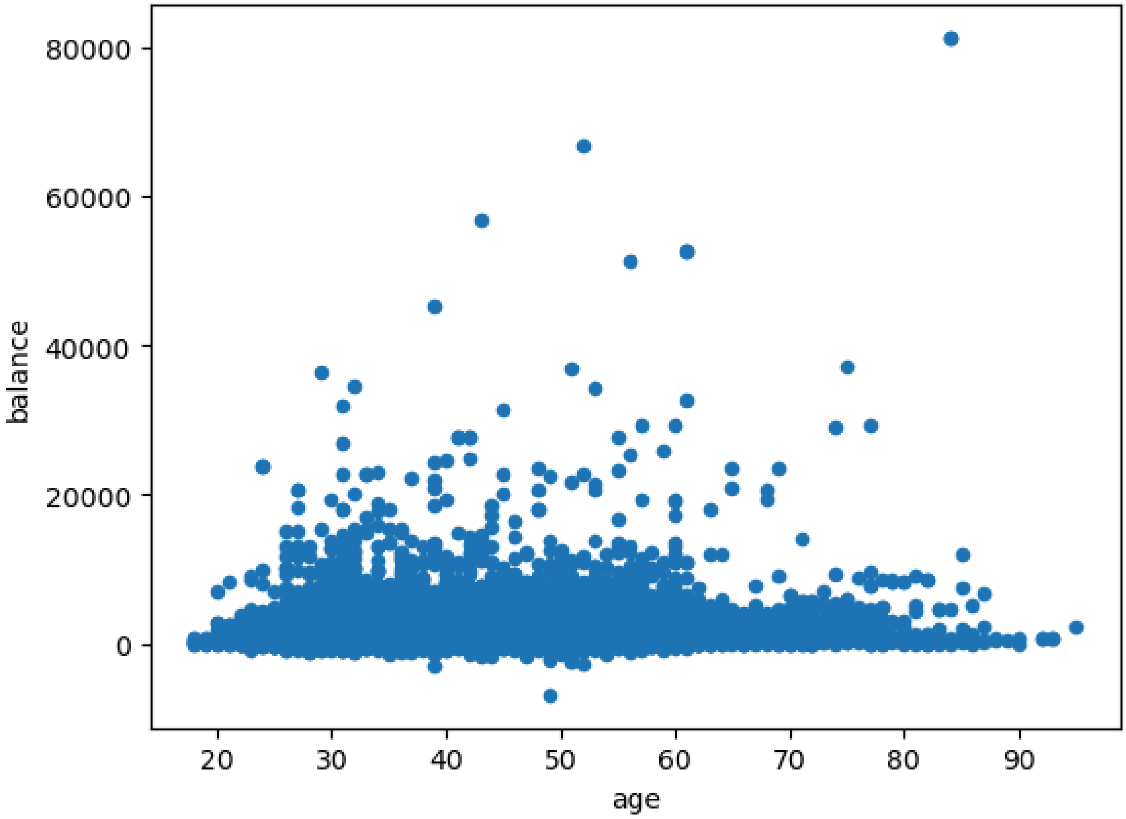
```
df.describe()
```

Out[17]:

	age	balance	day	duration	campaign	pdays
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000

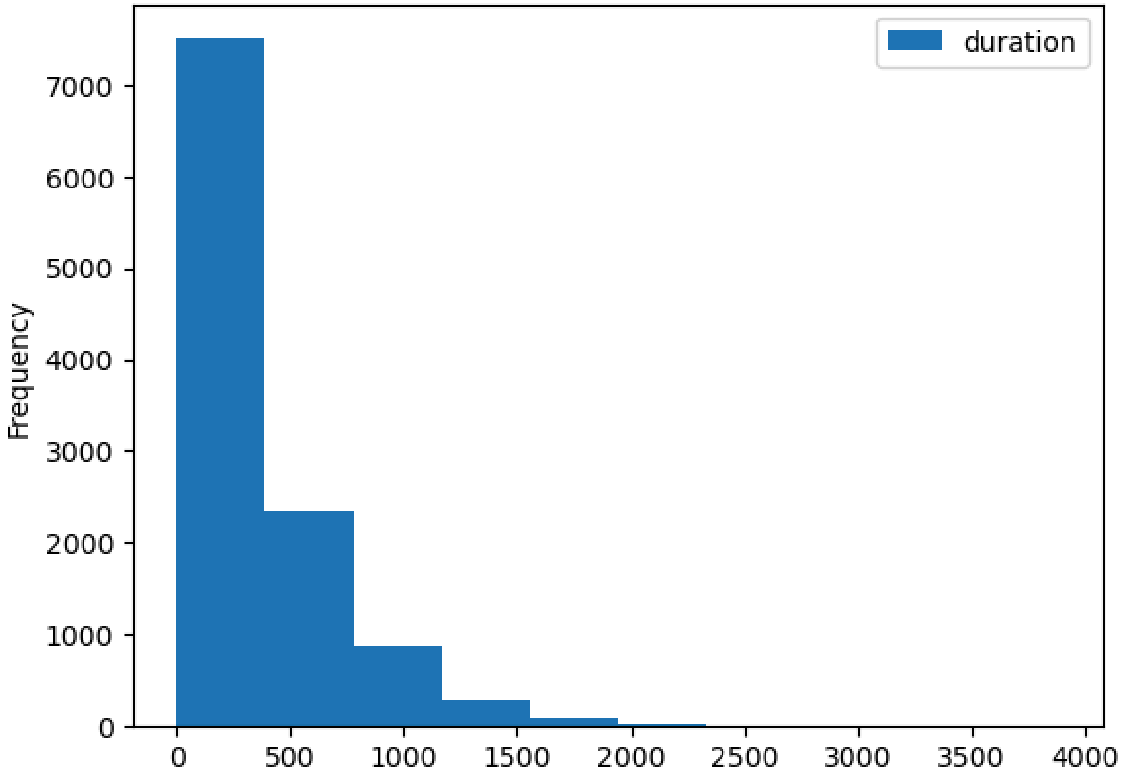
In [18]:

```
# Scatterplot showing age and balance
df.plot(kind='scatter', x='age', y='balance');

# Across all ages, majority of people have savings of less than 20000.
```



In [20]:

```
df.plot(kind='hist', x='poutcome', y='duration');
```



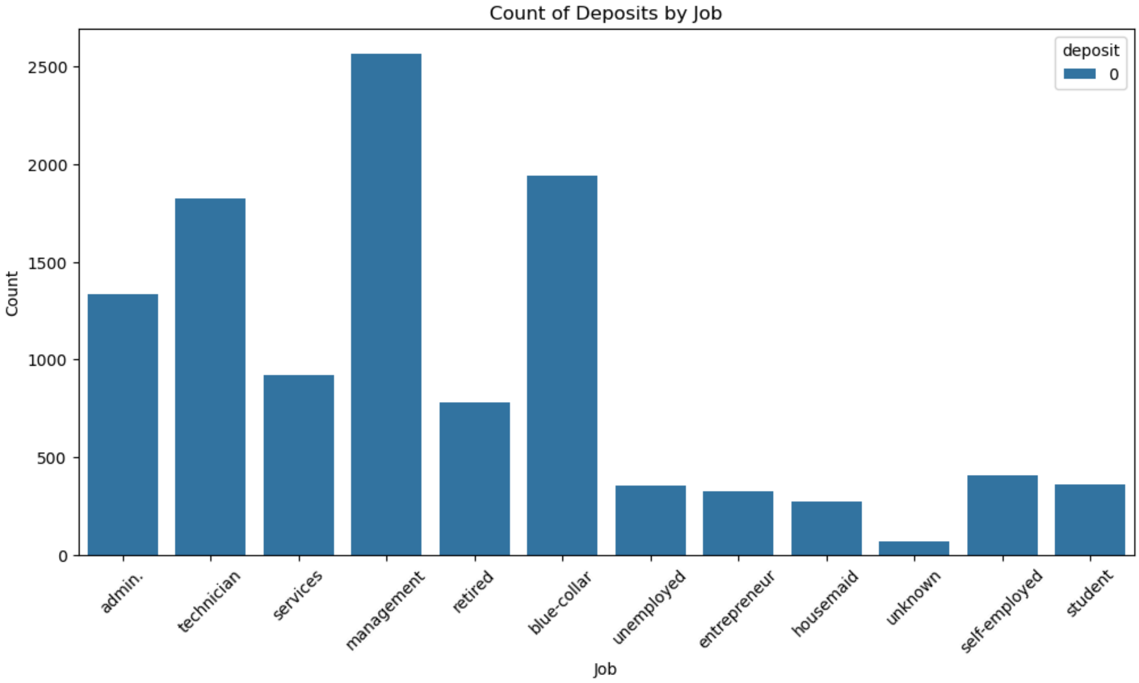
In [43]:

```
print(df['job'].value_counts())
print(df['deposit'].value_counts())
```

```
job
management       2566
blue-collar      1944
technician       1823
admin.           1334
services          923
retired           778
self-employed     405
student           360
unemployed        357
entrepreneur      328
housemaid         274
unknown            70
Name: count, dtype: int64
deposit
0    11162
Name: count, dtype: int64
```

In [45]:

```
# Count plot of job vs. deposit count

df['deposit'] = df['deposit'].apply(lambda x: 1 if x == 'yes' else 0)

plt.figure(figsize=(12, 6))
sns.countplot(x='job', hue='deposit', data=df)
plt.xticks(rotation=45)
plt.title('Count of Deposits by Job')
plt.xlabel('Job')
plt.ylabel('Count')
plt.show()
```



In [11]:
```
# Encode categorical variables using one-hot encoding
df_encoded = pd.get_dummies(df, drop_first=True)
df_encoded.head()
```

Out[11]:

	age	balance	day	duration	campaign	pdays	previous	job_blue-collar	job_entrepreneur	job_
0	59	2343	5	1042	1	-1	0	False	False	
1	56	45	5	1467	1	-1	0	False	False	
2	41	1270	5	1389	1	-1	0	False	False	
3	55	2476	5	579	1	-1	0	False	False	
4	54	184	5	673	2	-1	0	False	False	

5 rows × 43 columns

Train and Evaluate Decision Tree Classifier

In [6]:
```
# Define features (X) and target variable (y)
X = df_encoded.drop('deposit_yes', axis=1)
y = df_encoded['deposit_yes']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar
```

In [7]:
```
# Initialize the decision tree classifier
clf = DecisionTreeClassifier(random_state=42)

# Train the classifier
clf.fit(X_train, y_train)

# Predict the target variable on the test data
y_pred_dt = clf.predict(X_test)

# Calculate accuracy
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(f'Accuracy (Decision Tree): {accuracy_dt:.2f}')
```

```
Accuracy (Decision Tree): 0.79
```

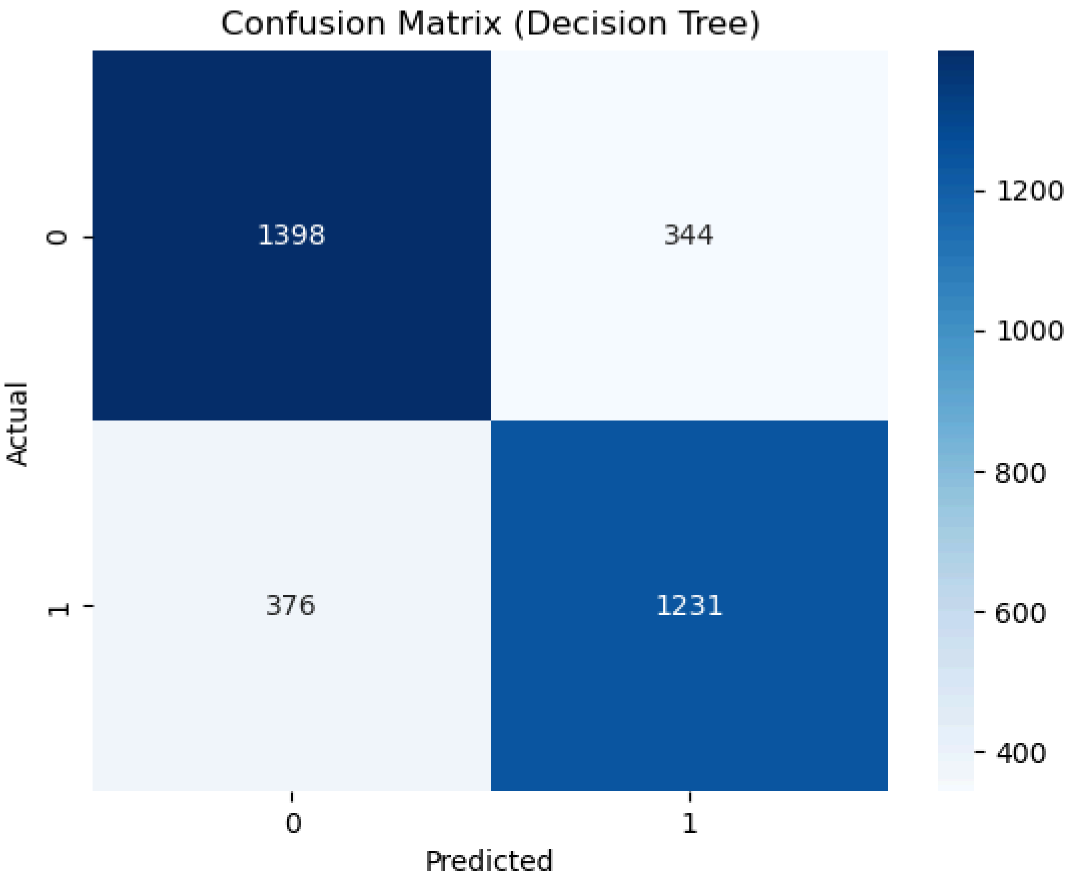
In [8]:

```
# Print classification report
print(classification_report(y_test, y_pred_dt))
```

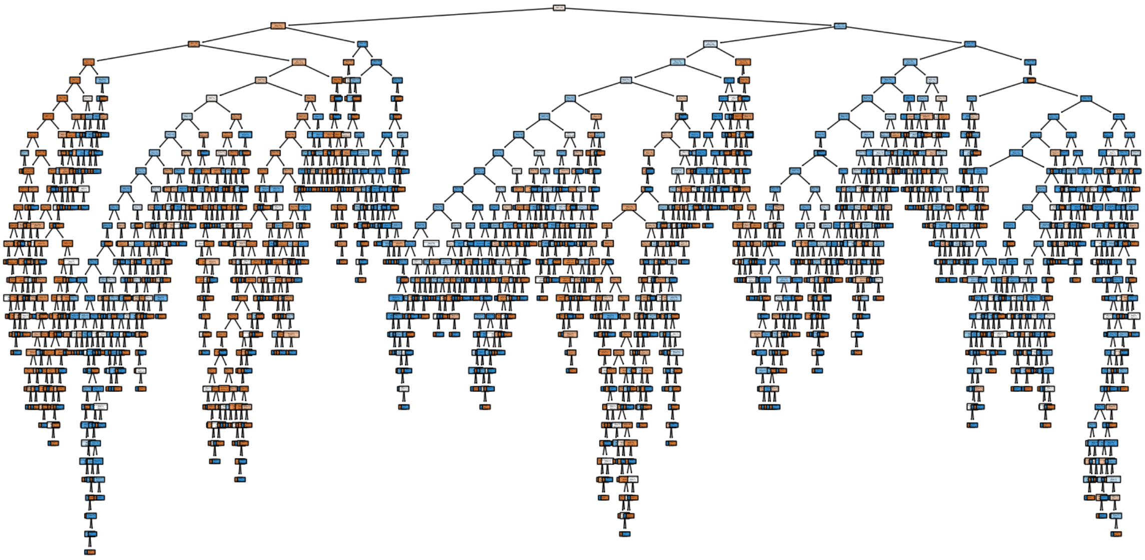
```
              precision    recall  f1-score   support

       False       0.79      0.80      0.80      1742
        True       0.78      0.77      0.77      1607

    accuracy                           0.79      3349
   macro avg       0.78      0.78      0.78      3349
weighted avg       0.78      0.79      0.78      3349
```

In [9]:

```
# Display confusion matrix
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(conf_matrix_dt, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Decision Tree)')
plt.show()
```



```
In [10]: # Plot the decision tree
         plt.figure(figsize=(20, 10))
         plot_tree(clf, filled=True, feature_names=list(X.columns), class_names=['No'
         plt.show()
```



In [14]:

```
new_customer = {
    'age': 65,
    'balance': 1200000,
    'day': 15,
    'duration': 300,
    'campaign': 1,
    'pdays': -1,
    'previous': 0,
    'job_blue-collar': 0,
    'job_entrepreneur': 0,
    'job_housemaid': 0,
    'job_management': 1,
    'job_retired': 0,
    'job_self-employed': 0,
    'job_services': 0,
    'job_student': 0,
    'job_technician': 0,
    'job_unemployed': 0,
    'job_unknown': 0,
    'marital_married': 1,
    'marital_single': 0,
    'education_secondary': 1,
    'education_tertiary': 0,
    'education_unknown': 0,
    'default_yes': 0,
    'housing_yes': 0,
    'loan_yes': 0,
    'contact_telephone': 0,
    'contact_unknown': 0,
    'month_aug': 0,
    'month_dec': 0,
    'month_feb': 0,
    'month_jan': 0,
    'month_jul': 0,
    'month_jun': 0,
    'month_mar': 0,
    'month_may': 1,
    'month_nov': 0,
    'month_oct': 0,
    'month_sep': 0,
    'poutcome_other': 0,
    'poutcome_success': 0,
    'poutcome_unknown': 1
}

# Convert the new customer data to a DataFrame
new_customer_df = pd.DataFrame([new_customer])
```

In [15]:

```
# Use the trained decision tree classifier to predict
new_customer_prediction = clf.predict(new_customer_df)

# Output the prediction
print(f'Prediction for new customer: {"Yes" if new_customer_prediction[0] ==
```

```
Prediction for new customer: Yes
```

In [ ]: