In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, Con
```

In [3]:
```
%pip install openpyxl
```

In [4]:
```
data=pd.read_excel("Bank.xlsx")
data.head(10)
```

Out[4]:

	age	marital	education	balance	housing	loan	day	month	duration_calls	campaign	subscribed
0	58	1	3	2143	1	0	5	5	261	1	0
1	44	2	2	29	1	0	5	5	151	1	0
2	33	1	2	2	1	1	5	5	76	1	0
3	47	1	4	1506	1	0	5	5	92	1	0
4	33	2	4	1	0	0	5	5	198	1	0
5	35	1	3	231	1	0	5	5	139	1	0
6	28	2	3	447	1	1	5	5	217	1	0
7	42	3	3	2	1	0	5	5	380	1	0
8	58	1	1	121	1	0	5	5	50	1	0
9	43	2	2	593	1	0	5	5	55	1	0

In [5]:
```
data.shape
```

Out[5]:
```
(45211, 11)
```

In [6]:
```
data.columns
```

Out[6]:
```
Index(['age', 'marital', 'education', 'balance', 'housing', 'loan', 'day',
       'month', 'duration_calls', 'campaign', 'subscribed'],
      dtype='object')
```

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

Out[7]:

	age	marital	education	balance	housing	loan	day
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.00000
mean	40.936210	1.513238	2.224813	1362.272058	0.555838	0.160226	15.80641
std	10.618762	0.692948	0.747997	3044.765829	0.496878	0.366820	8.32247
min	18.000000	1.000000	1.000000	-8019.000000	0.000000	0.000000	1.00000
25%	33.000000	1.000000	2.000000	72.000000	0.000000	0.000000	8.00000
50%	39.000000	1.000000	2.000000	448.000000	1.000000	0.000000	16.00000
75%	48.000000	2.000000	3.000000	1428.000000	1.000000	0.000000	21.00000
max	95.000000	3.000000	4.000000	102127.000000	1.000000	1.000000	31.00000

In [8]:
```
X=data.iloc[:,0:10]
X.head()
```

Out[8]:

	age	marital	education	balance	housing	loan	day	month	duration_calls	campaign
0	58	1	3	2143	1	0	5	5	261	1
1	44	2	2	29	1	0	5	5	151	1
2	33	1	2	2	1	1	5	5	76	1
3	47	1	4	1506	1	0	5	5	92	1
4	33	2	4	1	0	0	5	5	198	1

In [9]:
```
Y=data.iloc[:,10:11]
Y.head()
```

Out[9]:

	subscribed
0	0
1	0
2	0
3	0
4	0

In [10]:
```
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,test_size=0.2)
```

In [11]:
```
X_train.head()
```

Out[11]:

	age	marital	education	balance	housing	loan	day	month	duration_calls	campaign
30264	71	1	3	2651	0	0	5	2	531	2
14231	29	1	2	10	1	0	14	7	246	2
5791	27	2	2	162	0	0	26	5	265	9
13610	32	3	2	0	0	0	9	7	49	1
26171	57	3	2	3	0	0	20	11	77	3

In [12]:
```
tree=DecisionTreeClassifier(max_depth=3)
```

In [13]:
```
tree.fit(X_train.values, Y_train)
```

Out[13]:
```
▾ DecisionTreeClassifier
▶ Parameters
```

In [14]:
```
tree_score=tree.score(X_test.values, Y_test)
print("Accuracy:", tree_score)
```

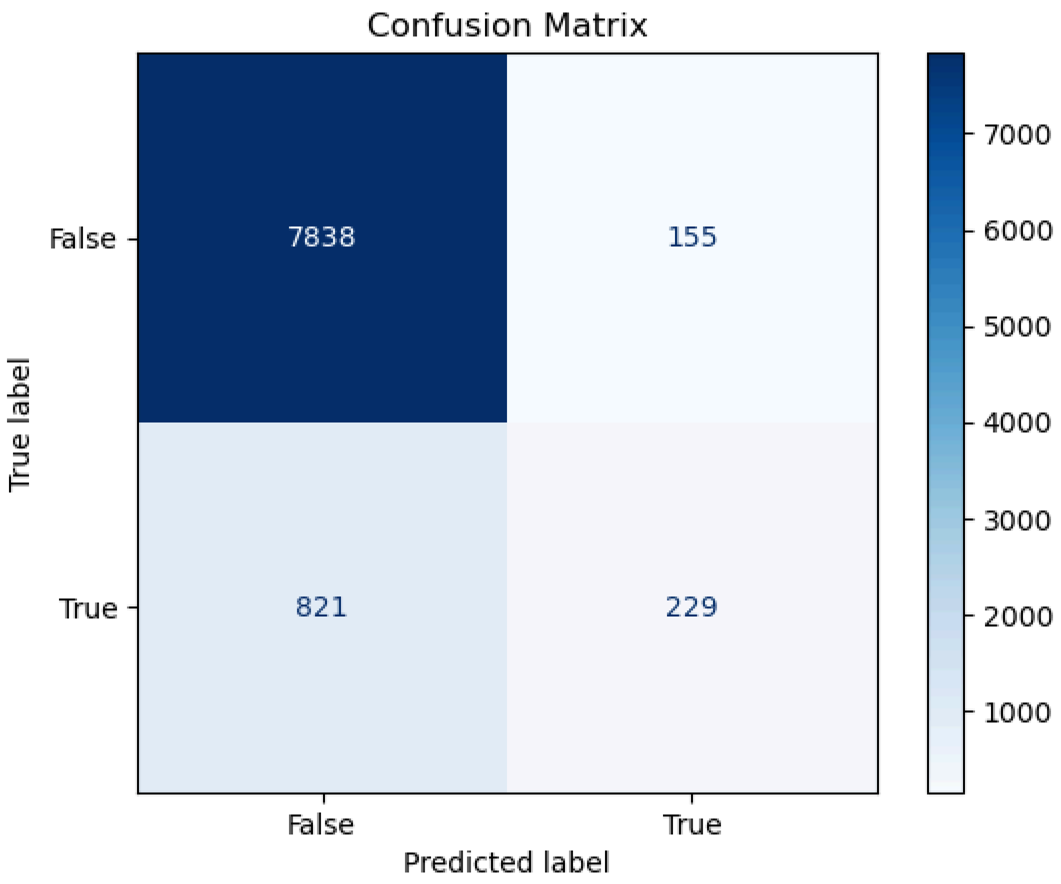
```
Accuracy: 0.8920712153046555
```

In [15]:
```
Y_pred = tree.predict(X_test.values)
```

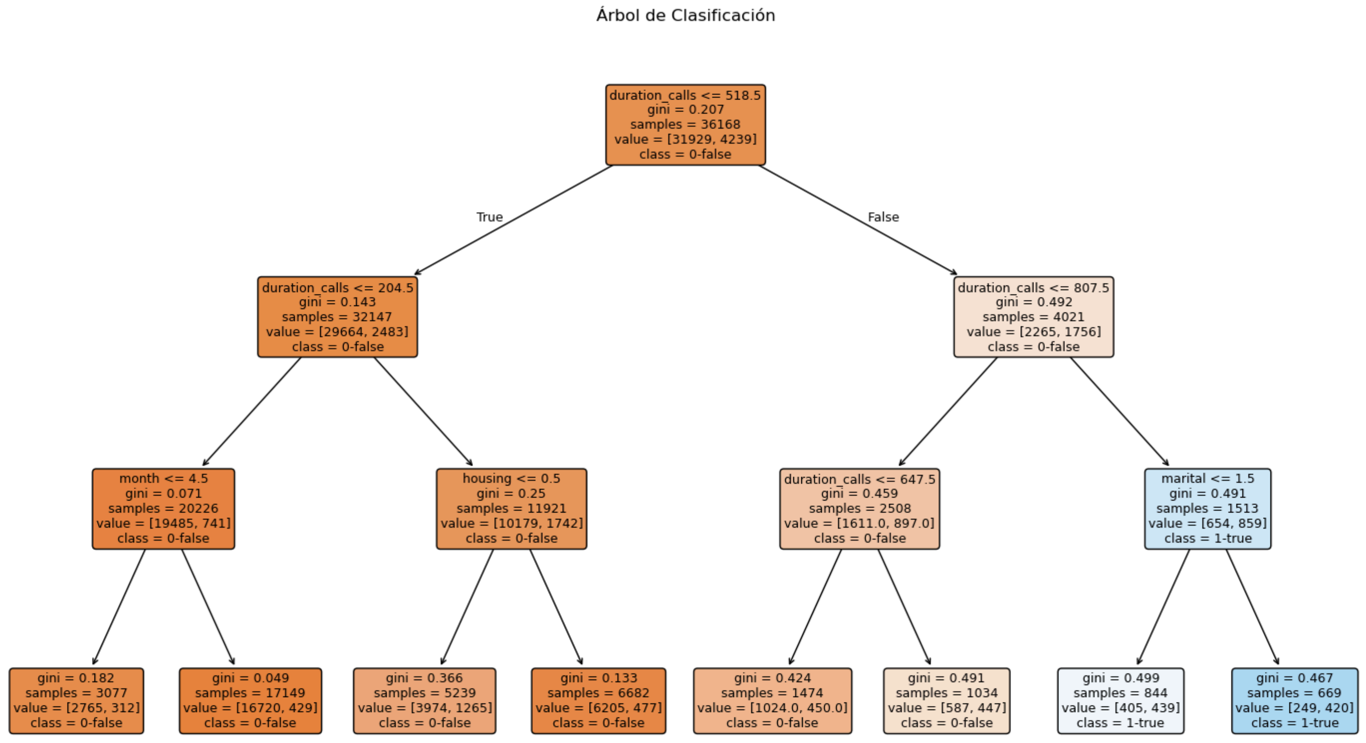
In [16]:
```
precision = precision_score(Y_test, Y_pred, pos_label=1)
recall = recall_score(Y_test, Y_pred, pos_label=1)

print("Precision:", round(precision,2))
print("Recall:", round(recall))
```

```
Precision: 0.6
Recall: 0
```

In [17]:
```
cm = confusion_matrix(Y_test, Y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["False", "True"])
disp.plot(cmap="Blues")
plt.title("Confusion Matrix")
plt.show()
```



```
In [18]: plt.figure(figsize=(18, 10))
         plot_tree(
             tree,
             feature_names=X.columns,
             class_names=["0-false", "1-true"],
             filled=True,
             rounded=True,
             fontsize=9
         )
         plt.title("Árbol de Clasificación")
         plt.show()
```


Árbol de Clasificación
duration_calls <= 518.5 gini = 0.207 samples = 36168 value = [31929, 4239] class = 0-false
True
False
duration_calls <= 204.5 gini = 0.143 samples = 32147 value = [29664, 2483] class = 0-false
duration_calls <= 807.5 gini = 0.492 samples = 4021 value = [2265, 1756] class = 0-false
month <= 4.5 gini = 0.071 samples = 20226 value = [19485, 741] class = 0-false
housing <= 0.5 gini = 0.25 samples = 11921 value = [10179, 1742] class = 0-false
duration_calls <= 647.5 gini = 0.459 samples = 2508 value = [1611.0, 897.0] class = 0-false
marital <= 1.5 gini = 0.491 samples = 1513 value = [654, 859] class = 1-true
gini = 0.182 samples = 3077 value = [2765, 312] class = 0-false
gini = 0.049 samples = 17149 value = [16720, 429] class = 0-false
gini = 0.366 samples = 5239 value = [3974, 1265] class = 0-false
gini = 0.133 samples = 6682 value = [6205, 477] class = 0-false
gini = 0.424 samples = 1474 value = [1024.0, 450.0] class = 0-false
gini = 0.491 samples = 1034 value = [587, 447] class = 0-false
gini = 0.499 samples = 844 value = [405, 439] class = 1-true
gini = 0.467 samples = 669 value = [249, 420] class = 1-true

In [19]:
```
importances = pd.Series(tree.feature_importances_, index=X.columns).sort_values(ascending=False)
importances
```

Out[19]:
```
duration_calls    0.852354
housing           0.120026
month             0.021508
marital           0.006112
age               0.000000
education         0.000000
balance           0.000000
loan              0.000000
day               0.000000
campaign          0.000000
dtype: float64
```

In [20]:
```
age=28
marital=1
education=3
balance=2000
housing=1
loan=1
day=2
month=10
duration_calls=1000
campaign=1

new_person=[age,marital,education,balance,housing,loan,day,month,duration_calls,campaign]
pred=tree.predict([new_person])
print("Al cliente se le asigna la clase: ",pred)
```

```
Al cliente se le asigna la clase:  [1]
```

In [21]:
```
age=28
marital=1
education=3
balance=2000
housing=1
```

```
loan=1
day=2
month=10
duration_calls=400
campaign=1

new_person=[age,marital,education,balance,housing,loan,day,month,duration_calls,campaign]
pred=tree.predict([new_person])
print("Al cliente se le asigna la clase: ",pred)
```

```
Al cliente se le asigna la clase:  [0]
```

Ejemplo 2: Multi clase

In [22]:
```
data=pd.read_excel("Product.xlsx")
data.head()
```

Out[22]:

	age	income_k	account_balance_k	num_contacts	months_as_client	interest_level
0	25	18	5	1	6	0
1	28	20	6	1	12	0
2	30	22	7	2	18	0
3	32	24	8	2	24	0
4	35	26	9	2	30	0

In [23]:
```
X=data.iloc[:,0:5]
X.head()
```

Out[23]:

	age	income_k	account_balance_k	num_contacts	months_as_client
0	25	18	5	1	6
1	28	20	6	1	12
2	30	22	7	2	18
3	32	24	8	2	24
4	35	26	9	2	30

In [24]:
```
Y=data.iloc[:,5:6]
Y.head()
```

Out[24]:

	interest_level
0	0
1	0
2	0
3	0
4	0

In [25]:
```
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,test_size=0.2)
```

In [26]:
```
X_train.head()
```

Out[26]:

	age	income_k	account_balance_k	num_contacts	months_as_client
15	52	40	20	4	80
42	26	31	16	2	45
39	31	37	21	3	60
31	48	57	37	6	98
37	34	41	24	4	69

In [27]:
```
tree=DecisionTreeClassifier(max_depth=3)
```

In [28]:
```
tree.fit(X_train.values, Y_train)
```

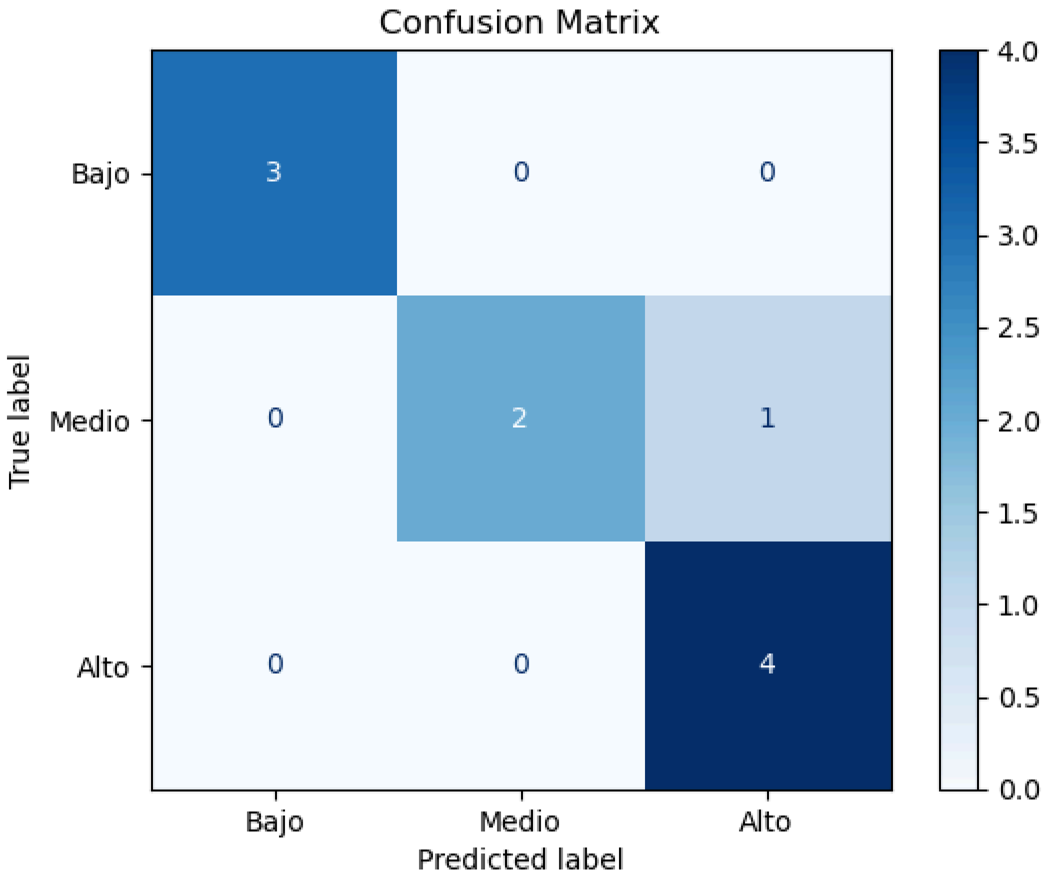
Out[28]:
```
▾ DecisionTreeClassifier

▶ Parameters
```

In [29]:
```
tree_score=tree.score(X_test.values, Y_test)
print("Accuracy:", tree_score)
```

```
Accuracy: 0.9
```

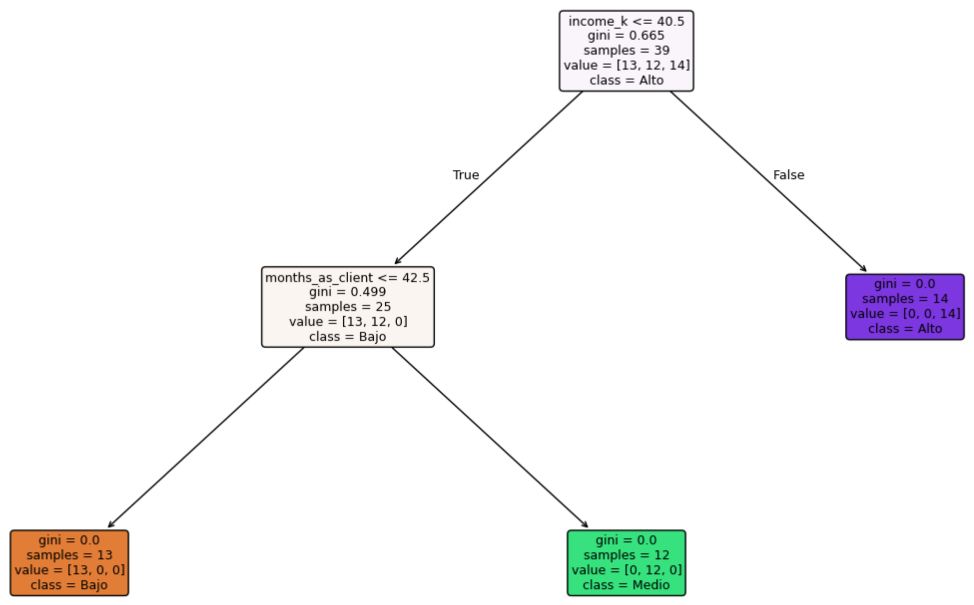
In [30]:
```
Y_pred = tree.predict(X_test.values)
```

In [31]:
```
cm = confusion_matrix(Y_test, Y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Bajo", "Medio", "Alto"])
disp.plot(cmap="Blues")
plt.title("Confusion Matrix")
plt.show()
```



In [32]:
```
plt.figure(figsize=(18, 10))
plot_tree(
    tree,
    feature_names=X.columns,
    class_names=["Bajo", "Medio", "Alto"],
    filled=True,
    rounded=True,
    fontsize=9
)
plt.title("Árbol de Clasificación")
plt.show()
```

Árbol de Clasificación



In [33]:
```
importances = pd.Series(tree.feature_importances_, index=X.columns).sort_values(ascending=False)
importances
```

Out[33]:
```
income_k             0.519051
months_as_client     0.480949
age                  0.000000
account_balance_k    0.000000
num_contacts         0.000000
dtype: float64
```

In [34]:
```
age= 28
income_k=20
account_balance_k=30
num_contacts=2
months_as_client=12
new=[age,income_k,account_balance_k,num_contacts,months_as_client]
predict=tree.predict([new])
print("El interés de comprar el producto para este cliente será: ",predict)
```

```
El interés de comprar el producto para este cliente será:  [0]
```

In [ ]: