In [3]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("Medical Appointment No Shows.csv")
```

In [4]:

```
df
```

Out[4]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA
...	...	...	...	...	...	...	...
110522	2.572134e+12	5651768	F	2016-05-03T09:15:35Z	2016-06-07T00:00:00Z	56	MARIA ORTIZ
110523	3.596266e+12	5650093	F	2016-05-03T07:27:33Z	2016-06-07T00:00:00Z	51	MARIA ORTIZ
110524	1.557663e+13	5630692	F	2016-04-27T16:03:52Z	2016-06-07T00:00:00Z	21	MARIA ORTIZ
110525	9.213493e+13	5630323	F	2016-04-27T15:09:23Z	2016-06-07T00:00:00Z	38	MARIA ORTIZ
110526	3.775115e+14	5629448	F	2016-04-27T13:30:56Z	2016-06-07T00:00:00Z	54	MARIA ORTIZ

110527 rows × 14 columns

In [7]:

```
df.isnull()
```

Out[7]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Sch
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Sch
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
110522	False	False	False	False	False	False	False	
110523	False	False	False	False	False	False	False	
110524	False	False	False	False	False	False	False	
110525	False	False	False	False	False	False	False	
110526	False	False	False	False	False	False	False	

110527 rows × 14 columns

In [9]:
```
df.drop_duplicates()
```
Out[9]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA
...	...	...	...	...	...	...	...
110522	2.572134e+12	5651768	F	2016-05-03T09:15:35Z	2016-06-07T00:00:00Z	56	MARIA ORTIZ
110523	3.596266e+12	5650093	F	2016-05-03T07:27:33Z	2016-06-07T00:00:00Z	51	MARIA ORTIZ
110524	1.557663e+13	5630692	F	2016-04-27T16:03:52Z	2016-06-07T00:00:00Z	21	MARIA ORTIZ
110525	9.213493e+13	5630323	F	2016-04-27T15:09:23Z	2016-06-07T00:00:00Z	38	MARIA ORTIZ
110526	3.775115e+14	5629448	F	2016-04-27T13:30:56Z	2016-06-07T00:00:00Z	54	MARIA ORTIZ

110527 rows × 14 columns

In [11]:
```
text_cols = ['PatientId','AppointmentID','Gender','ScheduledDay','Age','Neighbourhood','
def clean_text(series):
    return(
        series.astype(str)
        .str.strip()
        .str.lower()
        .str.replace(r'\s+',' ',regex=True)
    )
df[text_cols]=df[text_cols].apply(clean_text)
df[text_cols].head()
df
```

Out[11]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourh
0	29872499824296.0	5642903	f	2016-04-29t18:38:08z	2016-04-29T00:00:00Z	62	jardim da pe
1	558997776694438.0	5642503	m	2016-04-29t16:08:27z	2016-04-29T00:00:00Z	56	jardim da pe
2	4262962299951.0	5642549	f	2016-04-29t16:19:04z	2016-04-29T00:00:00Z	62	mata da p
3	867951213174.0	5642828	f	2016-04-29t17:29:31z	2016-04-29T00:00:00Z	8	ponta cam
4	8841186448183.0	5642494	f	2016-04-29t16:07:23z	2016-04-29T00:00:00Z	56	jardim da pe
...	...	...	...	...	...	...	
110522	2572134369293.0	5651768	f	2016-05-03t09:15:35z	2016-06-07T00:00:00Z	56	maria c
110523	3596266328735.0	5650093	f	2016-05-03t07:27:33z	2016-06-07T00:00:00Z	51	maria c
110524	15576631729893.0	5630692	f	2016-04-27t16:03:52z	2016-06-07T00:00:00Z	21	maria c
110525	92134931435557.0	5630323	f	2016-04-27t15:09:23z	2016-06-07T00:00:00Z	38	maria c
110526	377511518121127.0	5629448	f	2016-04-27t13:30:56z	2016-06-07T00:00:00Z	54	maria c

110527 rows × 14 columns

In [13]:

```
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'], errors='coerce', dayfirst=True)
df['ScheduledDay'] = df['ScheduledDay'].dt.strftime('%d-%m-%Y %H:%M:%S')
df['ScheduledDay']
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'], errors='coerce', dayfirst=Tr
df['AppointmentDay'] = df['AppointmentDay'].dt.strftime('%d-%m-%Y %H:%M:%S')
df['AppointmentDay']
df
```

```
C:\Users\ssneh\AppData\Local\Temp\ipykernel_20120\692614233.py:1: UserWarning: Parsing d
ates in %Y-%m-%dt%H:%M:%Sz format when dayfirst=True was specified. Pass `dayfirst=False
` or specify a format to silence this warning.
  df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'], errors='coerce', dayfirst=Tru
e)
C:\Users\ssneh\AppData\Local\Temp\ipykernel_20120\692614233.py:4: UserWarning: Parsing d
```

```
ates in %Y-%m-%dT%H:%M:%S%z format when dayfirst=True was specified. Pass `dayfirst=Fals
e` or specify a format to silence this warning.
  df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'], errors='coerce', dayfirst=
True)
```

Out[13]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourh
0	29872499824296.0	5642903	f	29-04-2016 18:38:08	29-04-2016 00:00:00	62	jardim da pe
1	558997776694438.0	5642503	m	29-04-2016 16:08:27	29-04-2016 00:00:00	56	jardim da pe
2	4262962299951.0	5642549	f	29-04-2016 16:19:04	29-04-2016 00:00:00	62	mata da p
3	867951213174.0	5642828	f	29-04-2016 17:29:31	29-04-2016 00:00:00	8	ponta cam
4	8841186448183.0	5642494	f	29-04-2016 16:07:23	29-04-2016 00:00:00	56	jardim da pe
...	...	...	...	...	...	...	
110522	2572134369293.0	5651768	f	03-05-2016 09:15:35	07-06-2016 00:00:00	56	maria c
110523	3596266328735.0	5650093	f	03-05-2016 07:27:33	07-06-2016 00:00:00	51	maria c
110524	15576631729893.0	5630692	f	27-04-2016 16:03:52	07-06-2016 00:00:00	21	maria c
110525	92134931435557.0	5630323	f	27-04-2016 15:09:23	07-06-2016 00:00:00	38	maria c
110526	377511518121127.0	5629448	f	27-04-2016 13:30:56	07-06-2016 00:00:00	54	maria c

110527 rows × 14 columns

In [53]:

```python
df.columns = (
    df.columns
    .str.strip()
    .str.lower()
    .str.replace(' ', '_')

)


df.columns.tolist()
df
```

Out[53]:

	gender	age	scholarship	hipertension	diabetes	alcoholism	handcap	sms_received	no-show	s
0	f	62	0	1	0	0	0	0	No	
1	m	56	0	0	0	0	0	0	No	

	gender	age	scholarship	hipertension	diabetes	alcoholism	handcap	sms_received	no-show
2	f	62	0	0	0	0	0	0	No
3	f	8	0	0	0	0	0	0	No
4	f	56	0	1	1	0	0	0	No
...	...	...	...	...	...	...	...	...	...
110522	f	56	0	0	0	0	0	1	No
110523	f	51	0	0	0	0	0	1	No
110524	f	21	0	0	0	0	0	1	No
110525	f	38	0	0	0	0	0	1	No
110526	f	54	0	0	0	0	0	1	No

110527 rows × 94 columns

In [17]:

```
df.dtypes
```

Out[17]:

```
patientid          object
appointmentid      object
gender             object
scheduledday       object
appointmentday     object
age                object
neighbourhood      object
scholarship        object
hipertension       object
diabetes           object
alcoholism         object
handcap            object
sms_received        int64
no-show            object
dtype: object
```

In [19]:

```
df['scheduledday'] = pd.to_datetime(df['scheduledday'], errors='coerce', dayfirst=True)
df['appointmentday']=pd.to_datetime(df['appointmentday'],errors='coerce',dayfirst=True)
df = df.astype({
    'age': 'int',
    'gender': 'category',
    'appointmentid':'int',
})
```

In [21]:

```
df.dtypes
```

Out[21]:

```
patientid                 object
appointmentid              int32
gender                  category
scheduledday      datetime64[ns]
appointmentday    datetime64[ns]
age                        int32
neighbourhood             object
scholarship               object
hipertension              object
diabetes                  object
alcoholism                object
handcap                   object
sms_received               int64
no-show                   object
dtype: object
```

In [23]:

```
X = df.drop(columns=['no-show', 'neighbourhood'])
```

In [25]:

```
X = df.drop(columns=['no-show', 'neighbourhood', 'scheduledday', 'appointmentday'])
```

In [27]:

```
df.columns = df.columns.str.strip().str.replace('\t', '', regex=False).str.replace('\n',


print(df.columns)


df['ScheduledDay'] = pd.to_datetime(df['scheduledday'], errors='coerce')
df['AppointmentDay'] = pd.to_datetime(df['appointmentday'], errors='coerce')


df = df.dropna(subset=['scheduledday', 'appointmentday'])


df['WaitingTime'] = (df['appointmentday'] - df['scheduledday']).dt.days


df['AppointmentWeekDay'] = df['appointmentday'].dt.day_name()
```

```
Index(['patientid', 'appointmentid', 'gender', 'scheduledday',
       'appointmentday', 'age', 'neighbourhood', 'scholarship', 'hipertension',
       'diabetes', 'alcoholism', 'handcap', 'sms_received', 'no-show'],
      dtype='object')
```

In [29]:

```
le_gender = LabelEncoder()
df['Gender'] = le_gender.fit_transform(df['gender'])

le_weekday = LabelEncoder()
df['AppointmentWeekDay'] = le_weekday.fit_transform(df['appointmentday'])
```

In [5]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

# Step 1: Load data
df = pd.read_csv('Medical Appointment No Shows.csv')

# Step 2: Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace('-', '_')

# Step 3: Drop rows with any missing data
df = df.dropna()

# Step 4: Convert date columns
df['scheduledday'] = pd.to_datetime(df['scheduledday'], errors='coerce')
df['appointmentday'] = pd.to_datetime(df['appointmentday'], errors='coerce')

# Drop rows where date conversion failed
df = df.dropna(subset=['scheduledday', 'appointmentday'])

# Step 5: Feature engineering
df['waiting_time'] = (df['appointmentday'] - df['scheduledday']).dt.days
df['appointment_weekday'] = df['appointmentday'].dt.dayofweek

# Step 6: Normalize and encode categorical variables
df['gender'] = df['gender'].str.strip().str.lower().map({'f': 0, 'm': 1})
df['no_show'] = df['no_show'].str.strip().str.lower().map({'no': 0, 'yes': 1})

# Drop rows where mapping failed
df = df.dropna(subset=['gender', 'no_show'])

# Encode 'neighbourhood' using one-hot encoding
df = pd.get_dummies(df, columns=['neighbourhood'], drop_first=True)

# Step 7: Drop unnecessary columns
df = df.drop(['patientid', 'appointmentid', 'scheduledday', 'appointmentday'], axis=1, e

# Step 8: Split into features and target
X = df.drop('no_show', axis=1)
y = df['no_show']

# Step 9: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Step 10: Apply SMOTE
smote = SMOTE(random_state=42)
X_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)

# Step 11: Train model
clf = DecisionTreeClassifier(max_depth=5, class_weight='balanced', random_state=42)
clf.fit(X_train_sm, y_train_sm)

# Step 12: Evaluate
```

```
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
D:\anaconda\Lib\site-packages\sklearn\base.py:474: FutureWarning: `BaseEstimator._valida
te_data` is deprecated in 1.6 and will be removed in 1.7. Use `sklearn.utils.validation.
validate_data` instead. This function becomes public and is part of the scikit-learn dev
eloper API.
  warnings.warn(
              precision    recall  f1-score   support

           0       0.92      0.51      0.66     17642
           1       0.30      0.82      0.44      4464

    accuracy                           0.57     22106
   macro avg       0.61      0.67      0.55     22106
weighted avg       0.79      0.57      0.61     22106
```

In [16]:

```
new_predictions = clf.predict(X_test)
df_results = X_test.copy()
df_results['Actual'] = y_test
df_results['Predicted'] = new_predictions
print("\n Sample Predictions:\n", df_results[['Actual', 'Predicted']].head())
```

```
 Sample Predictions:
         Actual  Predicted
23937        1          0
99403        0          1
100162       0          1
63869        1          0
7668         0          0
```

In [18]:

```
df_results.to_csv("Predicted_NoShows.csv", index=False)
```

In [20]:

```
df.to_csv('cleaned_medical_appointments.csv', index=False)
```

In [22]:

```
import os
print(os.getcwd())
```

```
C:\Users\ssneh
```

In [ ]: