# HEALTHCARE APPOINTMENT NO-SHOW PREDICTION

```
# 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.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)  # Optional: hide future warnings

# Step 2: Load Dataset
df = pd.read_excel(r"C:\Users\K.s.Santhos\OneDrive\Desktop\healthappo.xlsx")

# Step 3: Preprocessing
df.columns = [col.strip().replace('-', '_').replace(' ', '_') for col in df.columns]
df['No_show'] = df['No_show'].map({'No': 0, 'Yes': 1})
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'], errors='coerce')
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'], errors='coerce')
df['Weekday'] = df['AppointmentDay'].dt.day_name()
df['AppointmentMonth'] = df['AppointmentDay'].dt.month
df['DaysGap'] = (df['AppointmentDay'] - df['ScheduledDay']).dt.days
df = df[df['Age'] >= 0]
df['AgeGroup'] = pd.cut(df['Age'], bins=[0, 17, 59, 100], labels=['Child', 'Adult', 'Senior'])

# Clean Gender column
df['Gender'] = df['Gender'].astype(str).str.lower().str.strip()
```

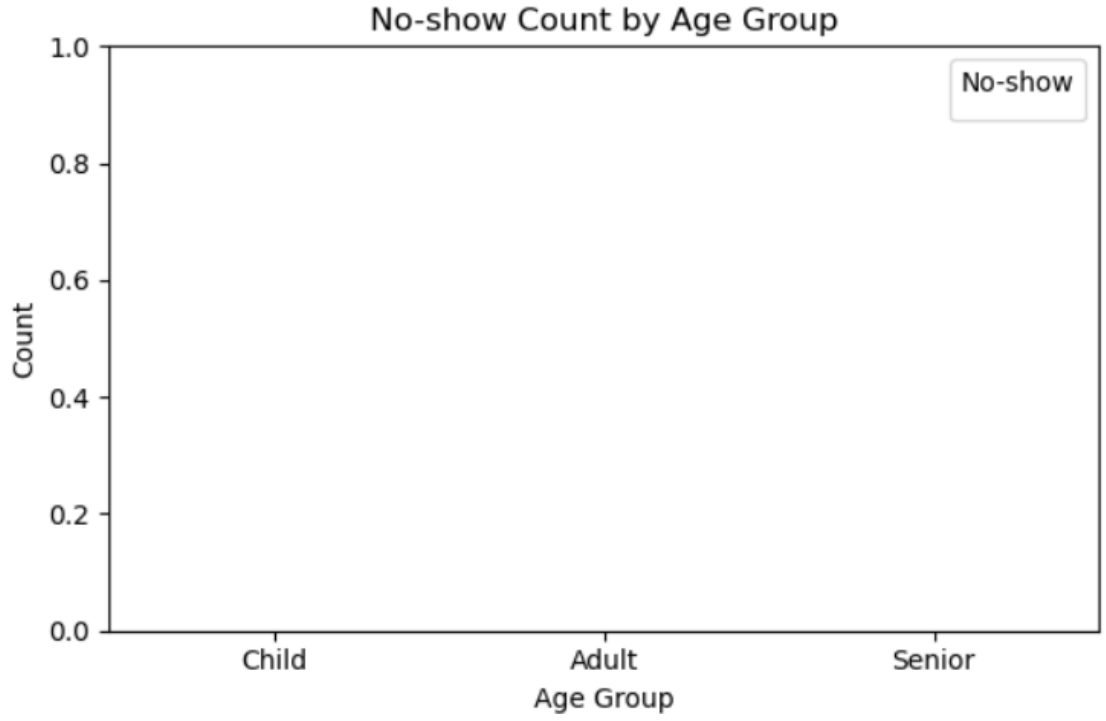
```
df['Gender'] = df['Gender'].replace({'female': 0, 'f': 0, 'male': 1, 'm':
1}).infer_objects(copy=False)


# Binary columns cleanup
binary_cols = ['Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', 'Handcap',
'SMS_received']
for col in binary_cols:
    df[col] = df[col].astype(str).str.strip().str.lower()
    df[col] = df[col].replace({'true': 1, 'false': 0, '1': 1, '0': 0}).infer_objects(copy=False)


# Step 4: Visualizations
# 1. No-show by Age Group
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='AgeGroup', hue='No_show')
plt.title("No-show Count by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Count")
plt.legend(title="No-show", labels=["No", "Yes"])
plt.tight_layout()
plt.show()


# 2. No-show by Weekday
plt.figure(figsize=(8, 4))
sns.countplot(data=df, x='Weekday', hue='No_show', order=[
        'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'])
plt.title("No-show by Appointment Weekday")
plt.xticks(rotation=45)
plt.legend(title="No-show", labels=["No", "Yes"])
plt.tight_layout()
plt.show()
```

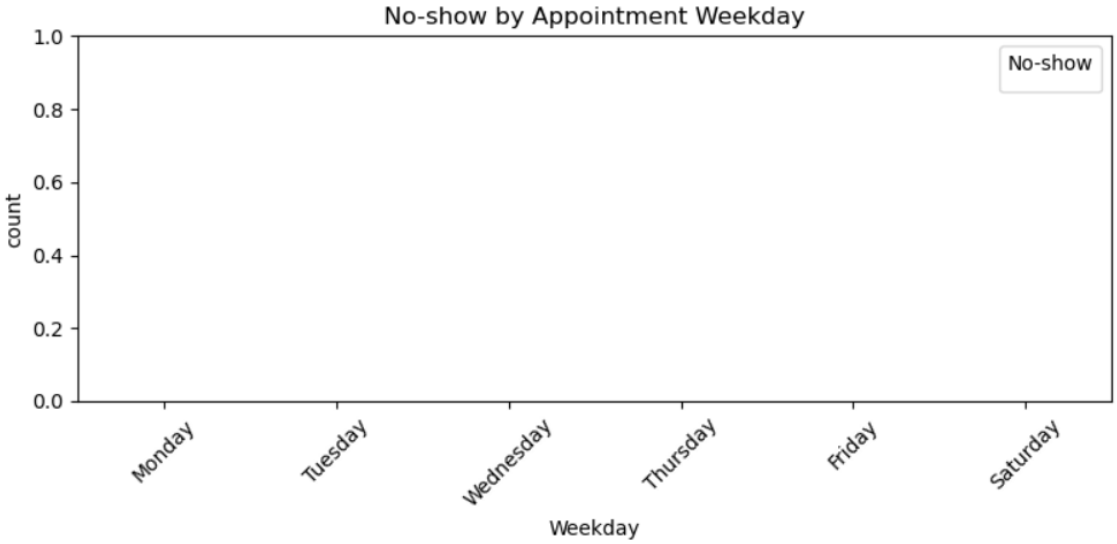
```
# 3. SMS Received Pie Chart
sms_counts = df['SMS_received'].value_counts().sort_index()
labels = ['No SMS', 'SMS Received']
sizes = [sms_counts.get(0, 0), sms_counts.get(1, 0)]
colors = ['#ff9999', '#66b3ff']
plt.figure(figsize=(5, 5))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('SMS Received Distribution')
plt.axis('equal')
plt.show()

# 4. Age Distribution
plt.figure(figsize=(6, 4))
sns.histplot(df['Age'], bins=30, kde=True)
plt.title("Age Distribution of Patients")
plt.xlabel("Age")
plt.ylabel("Number of Patients")
plt.tight_layout()
plt.show()

# 5. No-show by Gender
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='Gender', hue='No_show')
plt.title("No-show Count by Gender")
plt.xticks(ticks=[0, 1], labels=['Female', 'Male'])
plt.xlabel("Gender")
plt.ylabel("Count")
plt.legend(title="No-show", labels=["No", "Yes"])
plt.tight_layout()
plt.show()
```

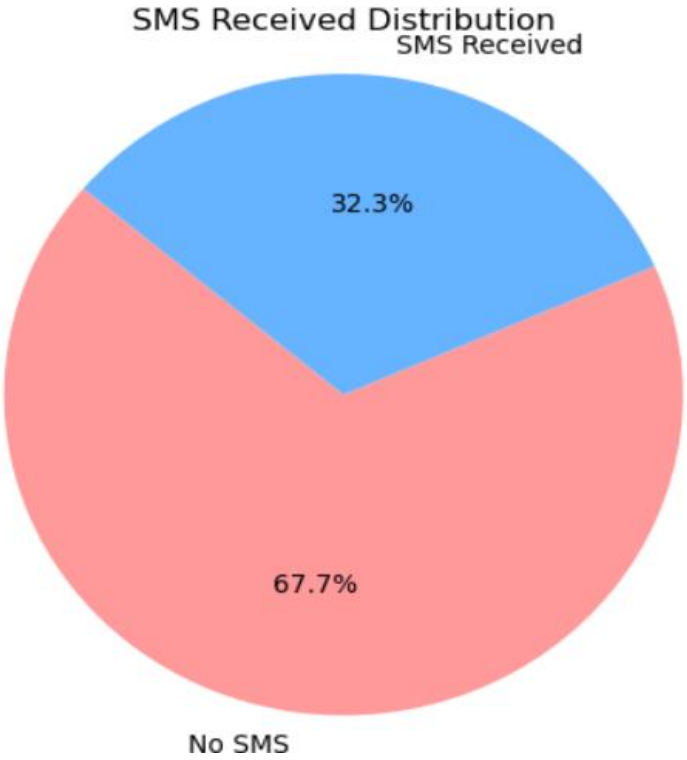
```
# 6. Top 10 Neighbourhoods with Most No-shows
top_neigh = df[df['No_show'] == 1]['Neighbourhood'].value_counts().nlargest(10)
plt.figure(figsize=(8, 4))
sns.barplot(x=top_neigh.values, y=top_neigh.index)
plt.title("Top 10 Neighbourhoods by No-show Count")
plt.xlabel("No-show Count")
plt.ylabel("Neighbourhood")
plt.tight_layout()
plt.show()

# Step 5: Modeling

# Feature selection
features = ['Age', 'Gender', 'Scholarship', 'Hipertension', 'Diabetes',
            'Alcoholism', 'Handcap', 'SMS_received', 'DaysGap']
X = df[features].apply(pd.to_numeric, errors='coerce').fillna(0)
y = df['No_show'].fillna(0)

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

# Decision Tree Model
model = DecisionTreeClassifier(max_depth=5, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Step 6: Evaluation
print("\n--- Model Evaluation ---")
```

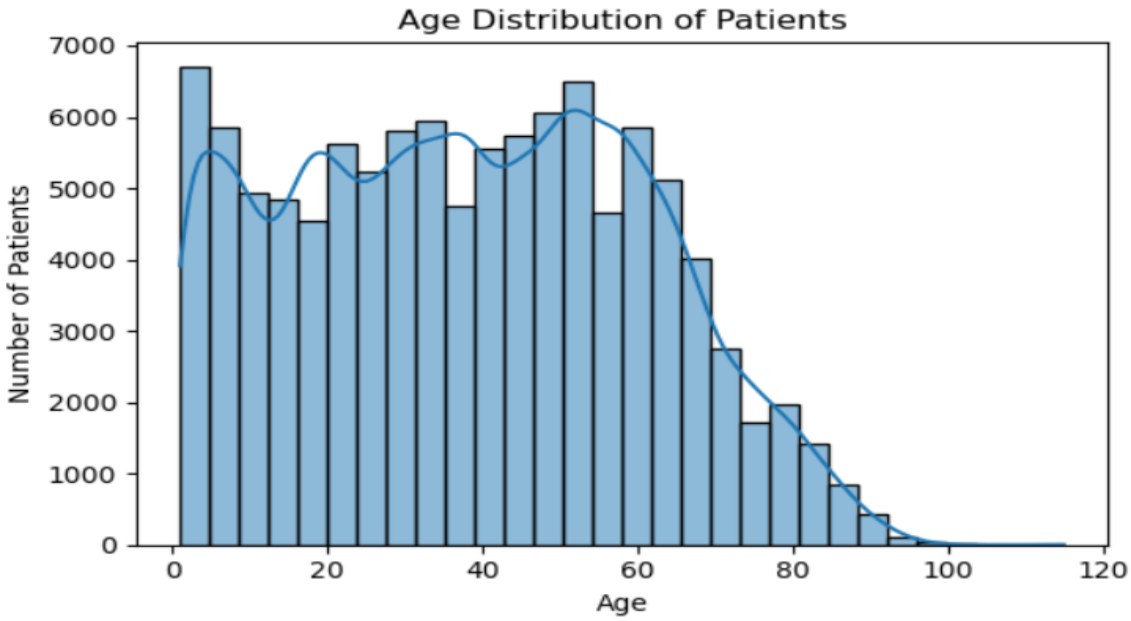
```
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred, labels=[0, 1]))

# Step 7: Feature Importance
plt.figure(figsize=(8, 4))
sns.barplot(x=model.feature_importances_, y=features)
plt.title("Feature Importance in No-show Prediction")
plt.xlabel("Importance")
plt.ylabel("Features")
plt.tight_layout()
plt.show()
```

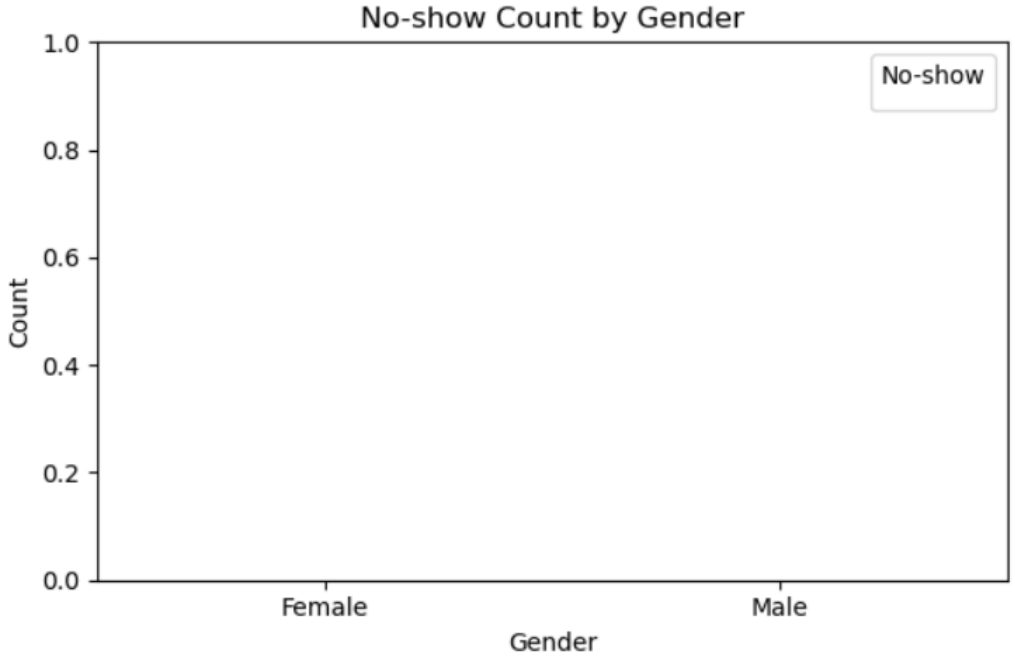
**OUTPUT:**

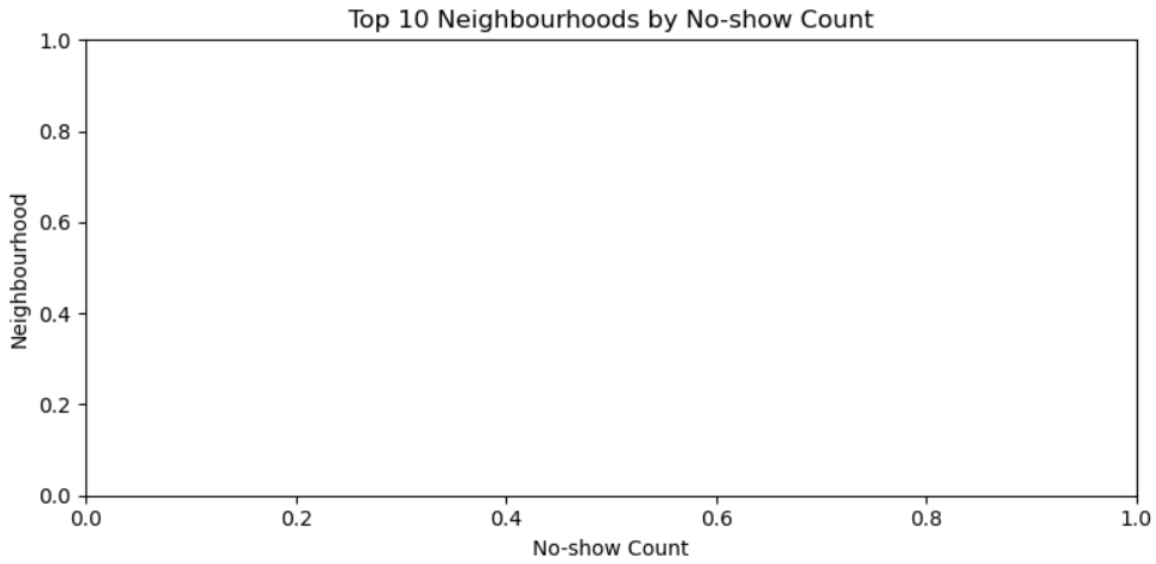



No-show by Appointment Weekday
No-show
count
1.0
0.8
0.6
0.4
0.2
0.0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Weekday


SMS Received Distribution
SMS Received
32.3%
67.7%
No SMS


Age Distribution of Patients
Number of Patients
Age
0
1000
2000
3000
4000
5000
6000
7000
0
20
40
60
80
100
120


No-show Count by Gender
No-show
Count
Gender
0.0
0.2
0.4
0.6
0.8
1.0
Female
Male




```
--- Model Evaluation ---
Accuracy Score: 1.0

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     21398

    accuracy                           1.00     21398
   macro avg       1.00      1.00      1.00     21398
weighted avg       1.00      1.00      1.00     21398

Confusion Matrix:
 [[21398     0]
 [    0     0]]
```