In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, classification_report, roc_auc_score, roc_curve
)


#Load Dataset
df =pd.read_csv('C:/Users/sarve/OneDrive/Desktop/Logistick Regression Healthcare Pr



#Preprocessing
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])
df['WaitTime'] = (df['AppointmentDay'] - df['ScheduledDay']).dt.days
df['No-show'] = df['No-show'].map({'No': 0, 'Yes': 1})


# data cleaning drop irrelivant columns

df = df.drop(columns = ['PatientId','AppointmentID'])



# Remove negative wait times
df = df[df['WaitTime'] >= 0]

# Encode categorical variables
df = pd.get_dummies(df, columns=['Gender', 'Neighbourhood'], drop_first=True)
```

In [2]:
```
df.head()
```

Out[2]:

	ScheduledDay	AppointmentDay	Age	Scholarship	Hipertension	Diabetes	Alcoholism
5	2016-04-27 08:36:51+00:00	2016-04-29 00:00:00+00:00	76	0	1	0	0
6	2016-04-27 15:05:12+00:00	2016-04-29 00:00:00+00:00	23	0	0	0	0
7	2016-04-27 15:39:58+00:00	2016-04-29 00:00:00+00:00	39	0	0	0	0
9	2016-04-27 12:48:25+00:00	2016-04-29 00:00:00+00:00	19	0	0	0	0
10	2016-04-27 14:58:11+00:00	2016-04-29 00:00:00+00:00	30	0	0	0	0

5 rows × 91 columns

In [3]:
```
#Feature and Target
X= df.drop(columns =['No-show', 'ScheduledDay', 'AppointmentDay'])
y =  df['No-show']
```

In [4]:
```
# Train and Test Split data

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_sta
```

In [5]:
```
#Feature scailing
scaler =StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

In [6]:
```
#Model Training
model = LogisticRegression(max_iter=1000)
model.fit(X_train_scaled,y_train)
```

Out[6]:

▾ LogisticRegression

```
LogisticRegression(max_iter=1000)
```

In [ ]:
```
#Predictions

y_pred = model.predict(X_test_scaled)

y_prob = model.predict_proba(X_test_scaled)[:,1]
```

Evaluating model performance

print("Accuracy:", round(accuracy_score(y_test, y_pred), 4)) print("Precision:", round(precision_score(y_test,y_pred),4)) print("Recall:", round(recall_score(y_test, y_pred), 4)) print("F1 Score:", round(f1_score(y_test, y_pred), 4)) print("ROC AUC:", round(roc_auc_score(y_test, y_prob), 4))

In [8]:
```
# Improving class imbalance with with class_weights
model =LogisticRegression(max_iter=1000,class_weight='balanced')
model.fit(X_train_scaled,y_train)
```

Out[8]:
```
LogisticRegression(class_weight='balanced', max_iter=1000)
```

In [9]:
```
#Threshold Tuning
threshold = 0.2  # example threshold, tune via validation
y_pred_custom = (y_prob >= threshold).astype(int)


print("Recall at threshold 0.2:", round(recall_score(y_test, y_pred_custom), 4))
print("Precision at threshold 0.2:", round(precision_score(y_test, y_pred_custom),
print("F1 Score at threshold 0.2:", round(f1_score(y_test, y_pred_custom), 4))
```

```
Recall at threshold 0.2: 0.9276
Precision at threshold 0.2: 0.2946
F1 Score at threshold 0.2: 0.4472
```

In [10]:
```
## Over Sampling the minority class using SMOTE
from imblearn.over_sampling import SMOTE
```

In [11]:
```
#using smote to resample the minority classes using synthetic resampling
sm =SMOTE(random_state=42)

#fit_resample function will perform synthetic resampling

X_train_res, y_train_res = sm.fit_resample(X_train_scaled, y_train)
```

In [12]:
```
### Training model again after performing the SMOTE
model =LogisticRegression(max_iter=1000)
model.fit(X_train_res,y_train_res)
```

Out[12]:
```
LogisticRegression(max_iter=1000)
```

In [13]:
```
## model prediction
y_pred = model.predict(X_test_scaled)
y_proba = model.predict_proba(X_test_scaled)[:, 1]
```

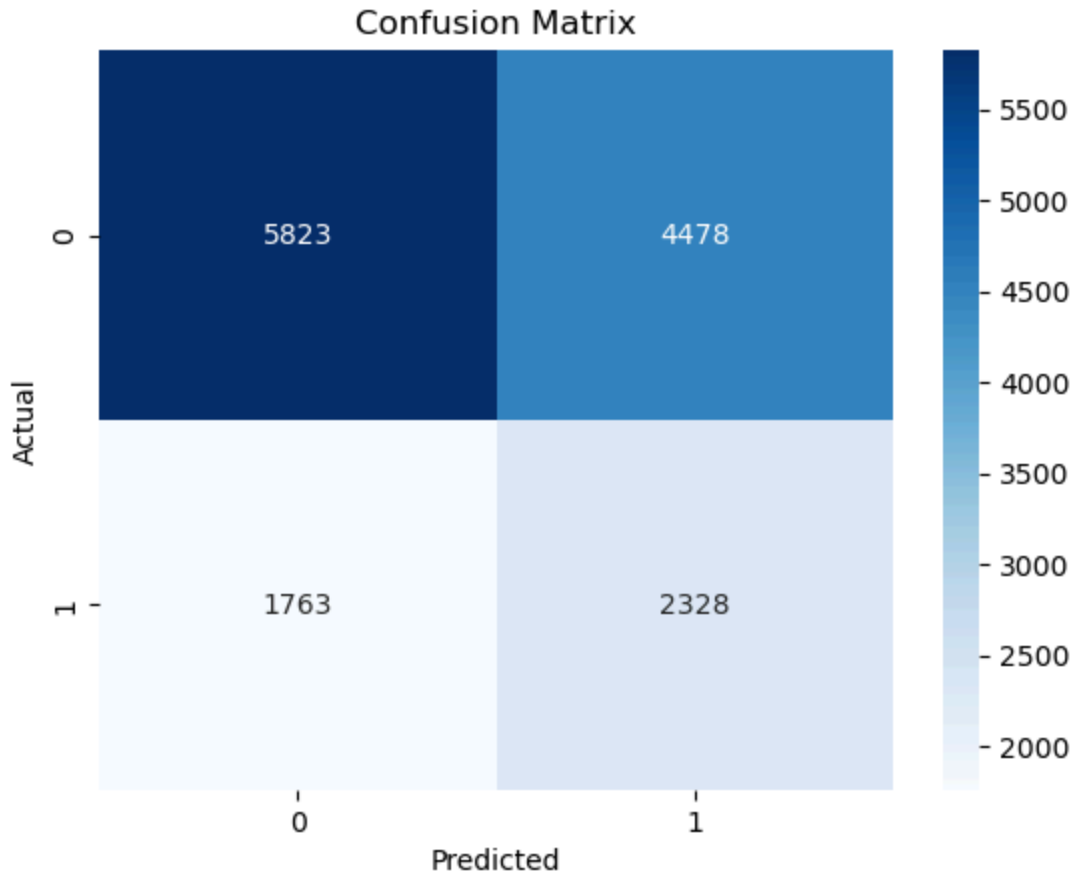
In [14]:
```
## Evaluating Model predictions

print("Recall after SMOTE:", round(recall_score(y_test, y_pred), 4))
```

```
print("Precision after SMOTE:", round(precision_score(y_test, y_pred), 4))
print("F1 Score after SMOTE:", round(f1_score(y_test, y_pred), 4))
print("ROC AUC:", round(roc_auc_score(y_test, y_prob), 4))
```

```
Recall after SMOTE: 0.5691
Precision after SMOTE: 0.3421
F1 Score after SMOTE: 0.4273
ROC AUC: 0.5888
```

In [15]:
```
## Confision Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In [16]:
```
# Cross Validation to measure how well model would perform on unseen data
cv_scores = cross_val_score(model,X, y, cv=5, scoring='roc_auc')
print("Cross-validated ROC AUC scores:", np.round(cv_scores, 4))
```

```
Cross-validated ROC AUC scores: [0.5822 0.5789 0.5669 0.5611 0.5857]
```

In [17]:
```
print("Cross-validated ROC AUC scores:", np.round(cv_scores, 4))
```

```
Cross-validated ROC AUC scores: [0.5822 0.5789 0.5669 0.5611 0.5857]
```

In [18]:
```
print("Average CV ROC AUC:", round(np.mean(cv_scores), 4))
```

```
Average CV ROC AUC: 0.575
```

In [19]:
```
patient_results = X_test.copy()
```

In [20]:
```
patient_results["Predicted_Probability"] = y_prob
```

In [21]:
```
patient_results
```

Out[21]:

	Age	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap	SMS_received	Wa
734	7	0	0	0	0	0	1	
79659	77	0	0	0	0	0	1	
60687	34	0	0	0	0	0	0	
10620	41	0	0	0	0	0	0	
41582	61	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	
38562	7	0	0	0	0	0	1	
27660	55	0	1	1	0	0	1	
46560	76	0	1	0	0	0	1	
93780	63	0	1	0	1	0	0	
3200	59	0	1	0	0	0	1	

14392 rows × 89 columns

In [22]:
```
y_preds = model.predict(X_test)  # 0 or 1
patient_results["Predicted_Label"] = y_preds
```

```
c:\Users\sarve\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2732: UserWar
ning: X has feature names, but LogisticRegression was fitted without feature names
  warnings.warn(
```

In [23]:
```
patient_results
```

Out[23]:

	Age	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap	SMS_received	Wa
734	7	0	0	0	0	0	1	
79659	77	0	0	0	0	0	1	
60687	34	0	0	0	0	0	0	
10620	41	0	0	0	0	0	0	
41582	61	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	
38562	7	0	0	0	0	0	1	
27660	55	0	1	1	0	0	1	
46560	76	0	1	0	0	0	1	
93780	63	0	1	0	1	0	0	
3200	59	0	1	0	0	0	1	

14392 rows × 90 columns

In [ ]: