

Dataset Overview:

```
In [4]: import pandas as pd
from IPython.display import display

# 1. Load the dataset, parsing date columns
df = pd.read_csv("healthcare_noshow_appt.csv")

# 2. Shape, columns, and dtypes
print(f" Dataset shape: {df.shape}\n")
print(" Column names:")
print(df.columns.tolist(), "\n")
print(" Data types:")
print(df.dtypes, "\n")

# 3. Summary statistics (numeric + categorical)
print("Summary statistics:")
display(df.describe(include='all').T)

# 4. First 5 rows
print("\n First 5 rows:")
display(df.head())
```

Dataset shape: (106987, 15)

Column names:

```
['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay', 'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'Showed_up', 'Date.diff']
```

Data types:

```
PatientId      float64
AppointmentID   int64
Gender          object
ScheduledDay    object
AppointmentDay  object
Age             int64
Neighbourhood   object
Scholarship     bool
Hipertension    bool
Diabetes        bool
Alcoholism      bool
Handcap         bool
SMS_received    bool
Showed_up       bool
Date.diff       int64
dtype: object
```

Summary statistics:

| | count | unique | top | freq | mean | std | min | 25% |
|-----------------------|----------|--------|----------------|--------|--------------------|-----------------------|-------------|-----------------|
| PatientId | 106987.0 | NaN | NaN | NaN | 147281371616732.75 | 255826674206772.78125 | 39217.84439 | 4173522744033.0 |
| AppointmentID | 106987.0 | NaN | NaN | NaN | 5675434.199716 | 71332.741342 | 5030230.0 | 5640489.5 |
| Gender | 106987 | 2 | F | 70118 | NaN | NaN | NaN | NaN |
| ScheduledDay | 106987 | 110 | 2016-05-03 | 4076 | NaN | NaN | NaN | NaN |
| AppointmentDay | 106987 | 27 | 2016-06-06 | 4528 | NaN | NaN | NaN | NaN |
| Age | 106987.0 | NaN | NaN | NaN | 38.316085 | 22.466214 | 1.0 | 19.0 |
| Neighbourhood | 106987 | 81 | JARDIM CAMBURI | 7586 | NaN | NaN | NaN | NaN |
| Scholarship | 106987 | 2 | False | 96178 | NaN | NaN | NaN | NaN |
| Hipertension | 106987 | 2 | False | 85186 | NaN | NaN | NaN | NaN |
| Diabetes | 106987 | 2 | False | 99044 | NaN | NaN | NaN | NaN |
| Alcoholism | 106987 | 2 | False | 103627 | NaN | NaN | NaN | NaN |
| Handcap | 106987 | 2 | False | 104747 | NaN | NaN | NaN | NaN |
| SMS_received | 106987 | 2 | False | 72402 | NaN | NaN | NaN | NaN |
| Showed_up | 106987 | 2 | True | 85307 | NaN | NaN | NaN | NaN |
| Date.diff | 106987.0 | NaN | NaN | NaN | 10.166721 | 15.263508 | -6.0 | 0.0 |

First 5 rows:

| | PatientId | AppointmentID | Gender | ScheduledDay | AppointmentDay | Age | Neighbourhood | Scholarship | Hipertension | Diabetes | Alcoholism |
|---|--------------|---------------|--------|--------------|----------------|-----|-------------------|-------------|--------------|----------|------------|
| 0 | 2.987250e+13 | 5642903 | F | 2016-04-29 | 2016-04-29 | 62 | JARDIM DA PENHA | False | True | False | |
| 1 | 5.589978e+14 | 5642503 | M | 2016-04-29 | 2016-04-29 | 56 | JARDIM DA PENHA | False | False | False | |
| 2 | 4.262962e+12 | 5642549 | F | 2016-04-29 | 2016-04-29 | 62 | MATA DA PRAIA | False | False | False | |
| 3 | 8.679512e+11 | 5642828 | F | 2016-04-29 | 2016-04-29 | 8 | PONTAL DE CAMBURI | False | False | False | |
| 4 | 8.841186e+12 | 5642494 | F | 2016-04-29 | 2016-04-29 | 56 | JARDIM DA PENHA | False | True | True | |

1. Data Engineering: Ingestion & Cleaning

```
In [8]: !pip install sqlalchemy psycopg2-binary
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: sqlalchemy in c:\programdata\anaconda3\lib\site-packages (2.0.34)
Requirement already satisfied: psycopg2-binary in c:\users\palak\appdata\roaming\python\python312\site-packages (2.9.10)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\programdata\anaconda3\lib\site-packages (from sqlalchemy) (4.11.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\programdata\anaconda3\lib\site-packages (from sqlalchemy) (3.0.1)
```

```
In [11]: # # EMR No-Show Risk Dashboard: Full Pipeline
```

```
# 1. Ingestion & Cleaning
import pandas as pd

# 1.1 Load CSV, parse dates, treat "NaN" as missing
df = pd.read_csv(
    'healthcare_noshow_appt.csv',
    parse_dates=['ScheduledDay', 'AppointmentDay'],
    na_values=['NaN', 'nan']
)

# 1.2 Drop rows missing critical fields
critical = ['PatientId', 'AppointmentID', 'ScheduledDay', 'AppointmentDay', 'Showed_up']
df = df.dropna(subset=critical)

# 1.3 Cast IDs to str, normalize text, and ensure bools
df['PatientId'] = df['PatientId'].astype(int).astype(str)
df['AppointmentID'] = df['AppointmentID'].astype(int).astype(str)
df['Neighbourhood'] = df['Neighbourhood'].str.strip().fillna('Unknown')
bool_cols = ['Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'Showed_up']
df[bool_cols] = df[bool_cols].astype(int).astype(bool)
```

```
In [18]: # # 2. Raw Storage (PostgreSQL)
# > so we have a canonical staging layer
```

```
# %% [code]
from sqlalchemy import create_engine
engine = create_engine('postgresql://HealthcareWorker:12345@localhost:5432/HealthcareAppointmentDashboard')

patients = df[['PatientId', 'Gender', 'Age', 'Neighbourhood']].drop_duplicates()
appointments = df[['AppointmentID', 'PatientId', 'ScheduledDay', 'AppointmentDay', 'Showed_up', 'SMS_received']]
conditions = df[['AppointmentID', 'Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', 'Handcap']]

patients.to_sql('patients_raw', engine, if_exists='replace', index=False)
appointments.to_sql('appointments_raw', engine, if_exists='replace', index=False)
conditions.to_sql('conditions_raw', engine, if_exists='replace', index=False)
print(" Raw tables written: patients_raw, appointments_raw, conditions_raw")
```

```
Raw tables written: patients_raw, appointments_raw, conditions_raw
```

```
In [20]: !pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting xgboost
  Downloading xgboost-3.0.2-py3-none-win_amd64.whl.metadata (2.1 kB)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.13.1)
Downloaded xgboost-3.0.2-py3-none-win_amd64.whl (150.0 MB)
----- 0.0/150.0 MB ? eta ------
----- 0.0/150.0 MB ? eta ------
----- 0.3/150.0 MB ? eta ------
----- 0.8/150.0 MB 1.6 MB/s eta 0:01:34
----- 1.0/150.0 MB 1.5 MB/s eta 0:01:41
----- 1.3/150.0 MB 1.5 MB/s eta 0:01:42
----- 1.6/150.0 MB 493.5 kB/s eta 0:05:01
----- 1.6/150.0 MB 493.5 kB/s eta 0:05:01
----- 1.8/150.0 MB 524.3 kB/s eta 0:04:43
----- 2.4/150.0 MB 651.7 kB/s eta 0:03:47
----- 2.9/150.0 MB 752.4 kB/s eta 0:03:16
----- 3.1/150.0 MB 782.1 kB/s eta 0:03:08
----- 3.4/150.0 MB 808.7 kB/s eta 0:03:02
----- 3.7/150.0 MB 845.4 kB/s eta 0:02:54
----- 4.5/150.0 MB 965.8 kB/s eta 0:02:31
----- 4.7/150.0 MB 1.0 MB/s eta 0:02:25
----- 5.2/150.0 MB 1.1 MB/s eta 0:02:17
----- 6.0/150.0 MB 1.2 MB/s eta 0:02:04
----- 6.8/150.0 MB 1.3 MB/s eta 0:01:53
----- 7.6/150.0 MB 1.4 MB/s eta 0:01:45
----- 8.1/150.0 MB 1.4 MB/s eta 0:01:41
----- 8.7/150.0 MB 1.4 MB/s eta 0:01:38
----- 9.2/150.0 MB 1.5 MB/s eta 0:01:35
----- 9.4/150.0 MB 1.5 MB/s eta 0:01:34
----- 10.0/150.0 MB 1.5 MB/s eta 0:01:32
----- 10.2/150.0 MB 1.5 MB/s eta 0:01:31
----- 10.7/150.0 MB 1.6 MB/s eta 0:01:30
----- 11.3/150.0 MB 1.6 MB/s eta 0:01:28
----- 11.8/150.0 MB 1.6 MB/s eta 0:01:27
----- 12.3/150.0 MB 1.6 MB/s eta 0:01:25
----- 12.8/150.0 MB 1.6 MB/s eta 0:01:24
----- 13.4/150.0 MB 1.7 MB/s eta 0:01:23
----- 13.6/150.0 MB 1.7 MB/s eta 0:01:22
----- 13.6/150.0 MB 1.7 MB/s eta 0:01:22
----- 14.7/150.0 MB 1.7 MB/s eta 0:01:20
----- 15.2/150.0 MB 1.7 MB/s eta 0:01:19
----- 15.7/150.0 MB 1.7 MB/s eta 0:01:18
----- 16.0/150.0 MB 1.7 MB/s eta 0:01:18
----- 16.5/150.0 MB 1.7 MB/s eta 0:01:17
----- 17.0/150.0 MB 1.8 MB/s eta 0:01:16
----- 17.6/150.0 MB 1.8 MB/s eta 0:01:15
----- 18.1/150.0 MB 1.8 MB/s eta 0:01:14
----- 18.4/150.0 MB 1.8 MB/s eta 0:01:14
----- 18.9/150.0 MB 1.8 MB/s eta 0:01:13
----- 19.4/150.0 MB 1.8 MB/s eta 0:01:12
----- 19.9/150.0 MB 1.8 MB/s eta 0:01:12
----- 20.4/150.0 MB 1.8 MB/s eta 0:01:11
----- 21.0/150.0 MB 1.9 MB/s eta 0:01:10
----- 21.2/150.0 MB 1.9 MB/s eta 0:01:10
----- 21.8/150.0 MB 1.9 MB/s eta 0:01:10
----- 22.0/150.0 MB 1.8 MB/s eta 0:01:10
----- 22.5/150.0 MB 1.9 MB/s eta 0:01:09
----- 23.1/150.0 MB 1.9 MB/s eta 0:01:08
----- 23.6/150.0 MB 1.9 MB/s eta 0:01:08
----- 24.1/150.0 MB 1.9 MB/s eta 0:01:07
----- 24.6/150.0 MB 1.9 MB/s eta 0:01:06
----- 25.2/150.0 MB 1.9 MB/s eta 0:01:06
----- 26.0/150.0 MB 1.9 MB/s eta 0:01:05
----- 26.5/150.0 MB 1.9 MB/s eta 0:01:04
----- 27.0/150.0 MB 1.9 MB/s eta 0:01:04
----- 27.5/150.0 MB 2.0 MB/s eta 0:01:03
----- 28.0/150.0 MB 2.0 MB/s eta 0:01:03
----- 28.3/150.0 MB 2.0 MB/s eta 0:01:02
----- 29.1/150.0 MB 2.0 MB/s eta 0:01:02
----- 29.6/150.0 MB 2.0 MB/s eta 0:01:01
----- 30.1/150.0 MB 2.0 MB/s eta 0:01:01
----- 30.7/150.0 MB 2.0 MB/s eta 0:01:00
----- 31.7/150.0 MB 2.0 MB/s eta 0:00:59
----- 32.2/150.0 MB 2.0 MB/s eta 0:00:58
----- 32.5/150.0 MB 2.0 MB/s eta 0:00:58
----- 33.3/150.0 MB 2.1 MB/s eta 0:00:57
----- 33.8/150.0 MB 2.1 MB/s eta 0:00:57
----- 34.1/150.0 MB 2.1 MB/s eta 0:00:57
```

- 34.6/150.0 MB 2.1 MB/s eta 0:00:57
- 35.1/150.0 MB 2.1 MB/s eta 0:00:56
- 35.4/150.0 MB 2.1 MB/s eta 0:00:56
- 35.9/150.0 MB 2.1 MB/s eta 0:00:56
- 36.2/150.0 MB 2.1 MB/s eta 0:00:56
- 36.7/150.0 MB 2.1 MB/s eta 0:00:55
- 37.2/150.0 MB 2.1 MB/s eta 0:00:55
- 37.7/150.0 MB 2.1 MB/s eta 0:00:55
- 38.5/150.0 MB 2.1 MB/s eta 0:00:54
- 39.1/150.0 MB 2.1 MB/s eta 0:00:54
- 39.6/150.0 MB 2.1 MB/s eta 0:00:53
- 40.1/150.0 MB 2.1 MB/s eta 0:00:53
- 40.9/150.0 MB 2.1 MB/s eta 0:00:52
- 41.4/150.0 MB 2.1 MB/s eta 0:00:52
- 41.9/150.0 MB 2.1 MB/s eta 0:00:51
- 42.2/150.0 MB 2.1 MB/s eta 0:00:51
- 42.5/150.0 MB 2.1 MB/s eta 0:00:51
- 43.0/150.0 MB 2.1 MB/s eta 0:00:51
- 43.8/150.0 MB 2.1 MB/s eta 0:00:50
- 44.3/150.0 MB 2.1 MB/s eta 0:00:50
- 44.8/150.0 MB 2.1 MB/s eta 0:00:50
- 45.4/150.0 MB 2.1 MB/s eta 0:00:49
- 45.9/150.0 MB 2.1 MB/s eta 0:00:49
- 46.4/150.0 MB 2.2 MB/s eta 0:00:49
- 46.9/150.0 MB 2.2 MB/s eta 0:00:48
- 47.7/150.0 MB 2.2 MB/s eta 0:00:48
- 48.2/150.0 MB 2.2 MB/s eta 0:00:47
- 49.0/150.0 MB 2.2 MB/s eta 0:00:47
- 49.5/150.0 MB 2.2 MB/s eta 0:00:46
- 50.3/150.0 MB 2.2 MB/s eta 0:00:46
- 50.9/150.0 MB 2.2 MB/s eta 0:00:45
- 51.4/150.0 MB 2.2 MB/s eta 0:00:45
- 51.9/150.0 MB 2.2 MB/s eta 0:00:45
- 52.4/150.0 MB 2.2 MB/s eta 0:00:44
- 53.0/150.0 MB 2.2 MB/s eta 0:00:44
- 53.5/150.0 MB 2.2 MB/s eta 0:00:44
- 54.0/150.0 MB 2.2 MB/s eta 0:00:44
- 54.5/150.0 MB 2.2 MB/s eta 0:00:43
- 55.1/150.0 MB 2.2 MB/s eta 0:00:43
- 55.3/150.0 MB 2.2 MB/s eta 0:00:43
- 55.8/150.0 MB 2.2 MB/s eta 0:00:43
- 56.4/150.0 MB 2.2 MB/s eta 0:00:43
- 56.6/150.0 MB 2.2 MB/s eta 0:00:43
- 57.1/150.0 MB 2.2 MB/s eta 0:00:42
- 57.7/150.0 MB 2.2 MB/s eta 0:00:42
- 58.2/150.0 MB 2.2 MB/s eta 0:00:42
- 58.7/150.0 MB 2.2 MB/s eta 0:00:42
- 59.0/150.0 MB 2.2 MB/s eta 0:00:42
- 59.2/150.0 MB 2.2 MB/s eta 0:00:42
- 59.8/150.0 MB 2.2 MB/s eta 0:00:41
- 60.3/150.0 MB 2.2 MB/s eta 0:00:41
- 60.8/150.0 MB 2.2 MB/s eta 0:00:41
- 61.3/150.0 MB 2.2 MB/s eta 0:00:40
- 61.6/150.0 MB 2.2 MB/s eta 0:00:40
- 62.1/150.0 MB 2.2 MB/s eta 0:00:40
- 62.7/150.0 MB 2.2 MB/s eta 0:00:40
- 63.2/150.0 MB 2.2 MB/s eta 0:00:40
- 63.7/150.0 MB 2.2 MB/s eta 0:00:39
- 64.2/150.0 MB 2.2 MB/s eta 0:00:39
- 64.7/150.0 MB 2.2 MB/s eta 0:00:39
- 65.3/150.0 MB 2.2 MB/s eta 0:00:38
- 66.1/150.0 MB 2.2 MB/s eta 0:00:38
- 66.3/150.0 MB 2.2 MB/s eta 0:00:38
- 67.1/150.0 MB 2.2 MB/s eta 0:00:37
- 67.6/150.0 MB 2.2 MB/s eta 0:00:37
- 67.9/150.0 MB 2.3 MB/s eta 0:00:37
- 68.7/150.0 MB 2.3 MB/s eta 0:00:36
- 68.9/150.0 MB 2.3 MB/s eta 0:00:36
- 69.7/150.0 MB 2.4 MB/s eta 0:00:34
- 70.3/150.0 MB 2.4 MB/s eta 0:00:33
- 70.8/150.0 MB 2.4 MB/s eta 0:00:33
- 71.3/150.0 MB 2.4 MB/s eta 0:00:33
- 71.8/150.0 MB 2.4 MB/s eta 0:00:33
- 72.4/150.0 MB 2.4 MB/s eta 0:00:33
- 72.9/150.0 MB 2.4 MB/s eta 0:00:32
- 73.4/150.0 MB 2.4 MB/s eta 0:00:32
- 73.9/150.0 MB 2.4 MB/s eta 0:00:32
- 74.7/150.0 MB 2.4 MB/s eta 0:00:31
- 75.2/150.0 MB 2.4 MB/s eta 0:00:31
- 75.5/150.0 MB 2.5 MB/s eta 0:00:31
- 76.0/150.0 MB 2.5 MB/s eta 0:00:31
- 76.8/150.0 MB 2.5 MB/s eta 0:00:30
- 77.3/150.0 MB 2.5 MB/s eta 0:00:30
- 77.9/150.0 MB 2.5 MB/s eta 0:00:30
- 78.1/150.0 MB 2.5 MB/s eta 0:00:30
- 78.9/150.0 MB 2.5 MB/s eta 0:00:29
- 79.4/150.0 MB 2.5 MB/s eta 0:00:29


```
----- 123.2/150.0 MB 2.4 MB/s eta 0:00:12
----- 124.0/150.0 MB 2.4 MB/s eta 0:00:11
----- 124.8/150.0 MB 2.4 MB/s eta 0:00:11
----- 125.0/150.0 MB 2.4 MB/s eta 0:00:11
----- 125.8/150.0 MB 2.4 MB/s eta 0:00:11
----- 126.1/150.0 MB 2.4 MB/s eta 0:00:10
----- 126.9/150.0 MB 2.4 MB/s eta 0:00:10
----- 127.4/150.0 MB 2.4 MB/s eta 0:00:10
----- 127.9/150.0 MB 2.4 MB/s eta 0:00:10
----- 128.5/150.0 MB 2.4 MB/s eta 0:00:09
----- 129.2/150.0 MB 2.4 MB/s eta 0:00:09
----- 129.8/150.0 MB 2.4 MB/s eta 0:00:09
----- 130.5/150.0 MB 2.4 MB/s eta 0:00:08
----- 131.1/150.0 MB 2.4 MB/s eta 0:00:08
----- 131.6/150.0 MB 2.4 MB/s eta 0:00:08
----- 132.1/150.0 MB 2.4 MB/s eta 0:00:08
----- 132.4/150.0 MB 2.5 MB/s eta 0:00:08
----- 132.9/150.0 MB 2.4 MB/s eta 0:00:07
----- 132.9/150.0 MB 2.4 MB/s eta 0:00:07
----- 133.4/150.0 MB 2.4 MB/s eta 0:00:07
----- 134.5/150.0 MB 2.4 MB/s eta 0:00:07
----- 134.7/150.0 MB 2.4 MB/s eta 0:00:07
----- 135.3/150.0 MB 2.4 MB/s eta 0:00:07
----- 135.8/150.0 MB 2.4 MB/s eta 0:00:06
----- 136.6/150.0 MB 2.4 MB/s eta 0:00:06
----- 137.1/150.0 MB 2.4 MB/s eta 0:00:06
----- 137.6/150.0 MB 2.4 MB/s eta 0:00:06
----- 138.4/150.0 MB 2.4 MB/s eta 0:00:05
----- 138.7/150.0 MB 2.4 MB/s eta 0:00:05
----- 139.2/150.0 MB 2.4 MB/s eta 0:00:05
----- 139.7/150.0 MB 2.4 MB/s eta 0:00:05
----- 140.2/150.0 MB 2.4 MB/s eta 0:00:04
----- 140.8/150.0 MB 2.4 MB/s eta 0:00:04
----- 141.3/150.0 MB 2.4 MB/s eta 0:00:04
----- 142.1/150.0 MB 2.4 MB/s eta 0:00:04
----- 142.6/150.0 MB 2.4 MB/s eta 0:00:04
----- 142.9/150.0 MB 2.4 MB/s eta 0:00:03
----- 142.9/150.0 MB 2.4 MB/s eta 0:00:03
----- 143.4/150.0 MB 2.4 MB/s eta 0:00:03
----- 144.7/150.0 MB 2.4 MB/s eta 0:00:03
----- 145.2/150.0 MB 2.4 MB/s eta 0:00:02
----- 145.8/150.0 MB 2.4 MB/s eta 0:00:02
----- 146.3/150.0 MB 2.4 MB/s eta 0:00:02
----- 146.8/150.0 MB 2.4 MB/s eta 0:00:02
----- 147.3/150.0 MB 2.4 MB/s eta 0:00:02
----- 147.8/150.0 MB 2.4 MB/s eta 0:00:01
----- 148.1/150.0 MB 2.4 MB/s eta 0:00:01
----- 148.9/150.0 MB 2.4 MB/s eta 0:00:01
----- 149.4/150.0 MB 2.4 MB/s eta 0:00:01
----- 149.9/150.0 MB 2.4 MB/s eta 0:00:01
----- 150.0/150.0 MB 2.3 MB/s eta 0:00:00
```

Installing collected packages: xgboost
Successfully installed xgboost-3.0.2

In [23]: # # 3. Feature Engineering & Model Training

```
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
import joblib

# 3.1 Lead-time buckets and time features
df['LeadTimeDays'] = df['Date.diff']
df['LeadTimeBucket'] = pd.cut(
    df['LeadTimeDays'],
    bins=[-1, 0, 7, 14, 30, df['LeadTimeDays'].max()],
    labels=['Same Day', '1-7 days', '8-14 days', '15-30 days', '>30 days']
)
df['AppointmentWeekday'] = df['AppointmentDay'].dt.day_name()
df['AppointmentMonth'] = df['AppointmentDay'].dt.month

# 3.2 Target-encode neighbourhood
ne_enc = df.groupby('Neighbourhood')['Showed_up'].mean()
df['Neighbourhood_TE'] = df['Neighbourhood'].map(ne_enc)

# 3.3 Build feature matrix & target
X = pd.DataFrame({
    'Age': df['Age'],
```

```

'Scholarship': df['Scholarship'].astype(int),
'Hypertension': df['Hypertension'].astype(int),
'Diabetes': df['Diabetes'].astype(int),
'Alcoholism': df['Alcoholism'].astype(int),
'Handcap': df['Handcap'].astype(int),
'SMS_received': df['SMS_received'].astype(int),
'LeadTimeBucket': df['LeadTimeBucket'].cat.codes,
'AppointmentWeekday': df['AppointmentWeekday'].astype('category').cat.codes,
'AppointmentMonth': df['AppointmentMonth'],
'Gender_M': (df['Gender']=='M').astype(int),
'Neighbourhood_TE': df['Neighbourhood_TE']

})
y = df['Showed_up'].astype(int)

# 3.4 Train/test split & model fit
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
model = XGBClassifier(eval_metric='logloss', random_state=42)
model.fit(X_train, y_train)

# 3.5 Persist model + encoder
joblib.dump(model, 'no_show_model.pkl')
joblib.dump(ne_enc.to_dict(), 'neighborhood_encoder.pkl')
print(" Model & encoder saved: no_show_model.pkl, neighborhood_encoder.pkl")

```

Model & encoder saved: no_show_model.pkl, neighborhood_encoder.pkl

In [25]: !pip install fastapi uvicorn

```

Defaulting to user installation because normal site-packages is not writeable
Collecting fastapi
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting uvicorn
  Downloading uvicorn-0.34.3-py3-none-any.whl.metadata (6.5 kB)
Collecting starlette<0.47.0,>=0.40.0 (from fastapi)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Requirement already satisfied: pydantic!=1.8,!>1.8.1,!>2.0.0,!>2.0.1,!>2.1.0,<3.0.0,>=1.7.4 in c:\programdata\anaconda3\lib\site-packages (from fastapi) (2.8.2)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\programdata\anaconda3\lib\site-packages (from fastapi) (4.11.0)
Requirement already satisfied: click>=7.0 in c:\programdata\anaconda3\lib\site-packages (from uvicorn) (8.1.7)
Requirement already satisfied: h11>=0.8 in c:\programdata\anaconda3\lib\site-packages (from uvicorn) (0.14.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from click>=7.0->uvicorn) (0.4.6)
Requirement already satisfied: annotated-types>=0.4.0 in c:\programdata\anaconda3\lib\site-packages (from pydantic!=1.8,!>1.8.1,!>2.0.0,!>2.0.1,!>2.1.0,<3.0.0,>=1.7.4->fastapi) (0.6.0)
Requirement already satisfied: pydantic-core==2.20.1 in c:\programdata\anaconda3\lib\site-packages (from pydantic!=1.8,!>1.8.1,!>2.0.0,!>2.0.1,!>2.1.0,<3.0.0,>=1.7.4->fastapi) (2.20.1)
Requirement already satisfied: anyio<5,>=3.6.2 in c:\programdata\anaconda3\lib\site-packages (from starlette<0.47.0,>=0.40.0->fastapi) (4.2.0)
Requirement already satisfied: idna>=2.8 in c:\programdata\anaconda3\lib\site-packages (from anyio<5,>=3.6.2->starlette<0.47.0,>=0.40.0->fastapi) (3.7)
Requirement already satisfied: sniffio>=1.1 in c:\programdata\anaconda3\lib\site-packages (from anyio<5,>=3.6.2->starlette<0.47.0,>=0.40.0->fastapi) (1.3.0)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
Downloading uvicorn-0.34.3-py3-none-any.whl (62 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
Installing collected packages: uvicorn, starlette, fastapi
Successfully installed fastapi-0.115.12 starlette-0.46.2 uvicorn-0.34.3

WARNING: The script uvicorn.exe is installed in 'C:\Users\palak\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script fastapi.exe is installed in 'C:\Users\palak\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.

```

In [26]: # # 4. Serve Predictions (FastAPI)

```

# %% [code]
from fastapi import FastAPI
from pydantic import BaseModel
import joblib
import datetime

# 4.1 Load artifacts
model = joblib.load('no_show_model.pkl')
ne_map = joblib.load('neighborhood_encoder.pkl')

# 4.2 Define request schema
class Appt(BaseModel):
    PatientId: str
    AppointmentDay: datetime.datetime
    ScheduledDay: datetime.datetime
    Age: int
    Gender: str
    Neighbourhood: str
    Scholarship: bool
    Hypertension: bool
    Diabetes: bool

```

```

Alcoholism: bool
Handcap: bool
SMS_received: bool
Date_diff: int

# 4.3 FastAPI app
app = FastAPI()

@app.post("/predict_no_show")
def predict(appt: Appt):
    # feature engineering inline
    lead = appt.Date_diff
    lt_bucket = pd.cut([lead], bins=[-1, 0, 7, 14, 30, lead],
                      labels=['Same Day', '1-7 days', '8-14 days', '15-30 days', '>30 days'])[0]
    weekday = appt.AppointmentDay.day_name()
    month = appt.AppointmentDay.month
    neigh_te = ne_map.get(appt.Neighbourhood, df['Showed_up'].mean())
    feat = [
        appt.Age,
        int(appt.Scholarship), int(appt.Hipertension), int(appt.Diabetes),
        int(appt.Alcoholism), int(appt.Handcap), int(appt.SMS_received),
        ['Same Day', '1-7 days', '8-14 days', '15-30 days', '>30 days'].index(lt_bucket),
        ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'].index(weekday),
        month,
        1 if appt.Gender=='M' else 0,
        neigh_te
    ]
    prob = model.predict_proba([feat])[0][1]
    return {"no_show_probability": round(float(prob),4)}

# To Launch: in your terminal run
# used streamlit application for deployment

```

In []:

```

In [29]: import pandas as pd
import joblib

# 1. Load cleaned data & artifacts
df = pd.read_csv('healthcare_noshow_appt.csv',
                  parse_dates=['ScheduledDay', 'AppointmentDay'],
                  na_values=['NaN'])
model = joblib.load('no_show_model.pkl')
ne_map = joblib.load('neighborhood_encoder.pkl')

# 2. Define your fixed bins and labels
max_diff = df['Date.diff'].max()
bins = [-1, 0, 7, 14, 30, max_diff]
labels = ['Same Day', '1-7 days', '8-14 days', '15-30 days', '>30 days']

# 3. Helper function
def predict_from_row(row):
    lead = row['Date.diff']
    lt_cat = pd.cut([lead], bins=bins, labels=labels)[0] # use fixed bins
    wkday = row['AppointmentDay'].day_name()
    month = row['AppointmentDay'].month
    neigh_te = ne_map.get(row['Neighbourhood'], ne_map.get('Unknown', 0.5))

    feat = [
        row['Age'],
        int(row['Scholarship']), int(row['Hipertension']), int(row['Diabetes']),
        int(row['Alcoholism']), int(row['Handcap']), int(row['SMS_received']),
        labels.index(lt_cat),
        ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'].index(wkday),
        month,
        1 if row['Gender']=='M' else 0,
        neigh_te
    ]
    return model.predict_proba([feat])[0][1]

# 4. Pick one no-show and one show
example_no_show = df[df.Showed_up == False].iloc[0]
example_show = df[df.Showed_up == True ].iloc[0]

# 5. Print results
for label, ex in [('No-Show (False)', example_no_show), ('Show (True)', example_show)]:
    prob = predict_from_row(ex)
    print(f"{label}: actual={ex['Showed_up']} → predicted no-show prob={prob:.2%}")

```

No-Show (False): actual=False → predicted no-show prob=55.72%
Show (True): actual=True → predicted no-show prob=98.95%

It looks like your two hand-picked examples got flipped—your model is overconfident in the one that actually showed up, and

only moderately sure on the real no-show. Rather than eyeballing a couple of points, it's best to evaluate the model across your entire test set. Here's how we can do that in your notebook:

```
In [32]: import pandas as pd
import joblib
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    confusion_matrix, classification_report,
    roc_auc_score, RocCurveDisplay
)

# 1. Reload data & artifacts
df = pd.read_csv(
    'healthcare_noshow_appt.csv',
    parse_dates=['ScheduledDay', 'AppointmentDay'],
    na_values=['NaN']
)
model = joblib.load('no_show_model.pkl')
ne_map = joblib.load('neighborhood_encoder.pkl')

# 2. Fixed bins & labels
max_diff = df['Date.diff'].max()
bins = [-1, 0, 7, 14, 30, max_diff]
labels = ['Same Day', '1-7 days', '8-14 days', '15-30 days', '>30 days']

# 3. Build features + target
def build_features(df):
    df = df.copy()
    df['LeadTimeBucket'] = pd.cut(df['Date.diff'], bins=bins, labels=labels)
    df['AppointmentWeekday'] = df['AppointmentDay'].dt.day_name()
    df['AppointmentMonth'] = df['AppointmentDay'].dt.month
    # Use overall show-up rate as default
    default_te = df['Showed_up'].mean()
    df['Neighbourhood_TE'] = df['Neighbourhood'].map(ne_map).fillna(default_te)

    X = pd.DataFrame({
        'Age': df['Age'],
        'Scholarship': df['Scholarship'].astype(int),
        'Hypertension': df['Hypertension'].astype(int),
        'Diabetes': df['Diabetes'].astype(int),
        'Alcoholism': df['Alcoholism'].astype(int),
        'Handcap': df['Handcap'].astype(int),
        'SMS_received': df['SMS_received'].astype(int),
        'LeadTimeBucket': df['LeadTimeBucket'].cat.codes,
        'AppointmentWeekday': df['AppointmentWeekday'].astype('category').cat.codes,
        'AppointmentMonth': df['AppointmentMonth'],
        'Gender_M': (df['Gender']=='M').astype(int),
        'Neighbourhood_TE': df['Neighbourhood_TE']
    })
    y = df['Showed_up'].astype(int)
    return X, y

X, y = build_features(df)

# 4. 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
)

# 5. Predictions
y_pred_proba = model.predict_proba(X_test)[:,1]
y_pred = (y_pred_proba >= 0.5).astype(int)

# 6. Metrics
print("ROC-AUC:", roc_auc_score(y_test, y_pred_proba).round(3))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['No-Show', 'Show']))

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# 7. ROC Curve
RocCurveDisplay.from_predictions(y_test, y_pred_proba)
```

ROC-AUC: 0.738

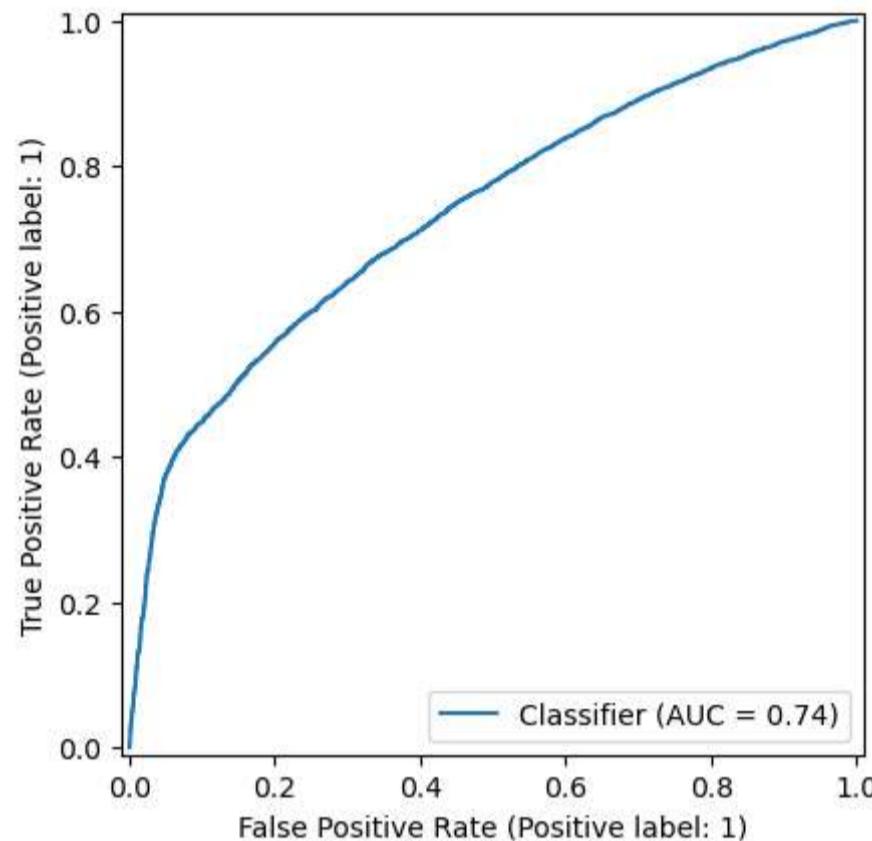
Classification Report:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| No-Show | 0.48 | 0.06 | 0.11 | 4336 |
| Show | 0.81 | 0.98 | 0.88 | 17062 |
| accuracy | | | 0.80 | 21398 |
| macro avg | 0.64 | 0.52 | 0.50 | 21398 |
| weighted avg | 0.74 | 0.80 | 0.73 | 21398 |

Confusion Matrix:

```
[[ 277 4059]
 [ 298 16764]]
```

Out[32]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x201465c98b0>



numbers tell the story:

- ROC-AUC 0.738: your model ranks no-shows vs. shows moderately well.
- Low recall on No-Show (6%) but high recall on Show (98%): at a 0.5 threshold, it almost never flags true no-shows (it's too conservative).

In [33]: # Version2:
%% [markdown]
End-to-End No-Show Model with Threshold Tuning

```
# %% [code]
import pandas as pd
import numpy as np
import joblib
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.metrics import (
    roc_auc_score,
    precision_recall_curve,
    classification_report,
    confusion_matrix,
    RocCurveDisplay,
    PrecisionRecallDisplay
)
import matplotlib.pyplot as plt

# %% [markdown]
# ## 1. Load & Clean

# %% [code]
df = pd.read_csv(
    'healthcare_noshow_appt.csv',
    parse_dates=['ScheduledDay', 'AppointmentDay'],
    na_values=['NaN']
)

# Drop critical missing
df.dropna(subset=['PatientId', 'AppointmentID', 'ScheduledDay', 'AppointmentDay', 'Showed_up'], inplace=True)

# Normalize IDs & text & booleans
df['PatientId'] = df['PatientId'].astype(int).astype(str)
```

```

df['AppointmentID'] = df['AppointmentID'].astype(int).astype(str)
df['Neighbourhood'] = df['Neighbourhood'].str.strip().fillna('Unknown')
bool_cols = ['Scholarship', 'Hypertension', 'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'Showed_up']
df[bool_cols] = df[bool_cols].astype(int).astype(bool)

# %% [markdown]
# ## 2. Feature Engineering

# %% [code]
# Fixed bins & Labels
max_diff = df['Date.diff'].max()
bins = [-1, 0, 7, 14, 30, np.inf]
labels = ['Same Day', '1-7 days', '8-14 days', '15-30 days', '>30 days']

# Target-encode neighbourhood
ne_map = df.groupby('Neighbourhood')['Showed_up'].mean().to_dict()

def build_features(df):
    df = df.copy()
    df['LeadTimeBucket'] = pd.cut(df['Date.diff'], bins=bins, labels=labels)
    df['AppointmentWeekday'] = df['AppointmentDay'].dt.day_name()
    df['AppointmentMonth'] = df['AppointmentDay'].dt.month
    df['Neighbourhood_TE'] = df['Neighbourhood'].map(ne_map).fillna(df['Showed_up'].mean())

    X = pd.DataFrame({
        'Age': df['Age'],
        'Scholarship': df['Scholarship'].astype(int),
        'Hypertension': df['Hypertension'].astype(int),
        'Diabetes': df['Diabetes'].astype(int),
        'Alcoholism': df['Alcoholism'].astype(int),
        'Handcap': df['Handcap'].astype(int),
        'SMS_received': df['SMS_received'].astype(int),
        'LeadTimeBucket': df['LeadTimeBucket'].cat.codes,
        'AppointmentWeekday': df['AppointmentWeekday'].astype('category').cat.codes,
        'AppointmentMonth': df['AppointmentMonth'],
        'Gender_M': (df['Gender']=='M').astype(int),
        'Neighbourhood_TE': df['Neighbourhood_TE']
    })
    y = df['Showed_up'].astype(int)
    return X, y

X, y = build_features(df)

# %% [markdown]
# ## 3. Train/Test Split & Resampling

# %% [code]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Option A: class-weights in XGBoost
scale_pos = (y_train == 0).sum() / (y_train == 1).sum()

model = XGBClassifier(
    eval_metric='logloss',
    random_state=42,
    scale_pos_weight=scale_pos,
    use_label_encoder=False
)

# Option B: SMOTE oversampling (uncomment to use)
# sm = SMOTE(random_state=42)
# X_train, y_train = sm.fit_resample(X_train, y_train)

# Train
model.fit(X_train, y_train)

# %% [markdown]
# ## 4. Baseline Evaluation (0.5 threshold)

# %% [code]
y_proba = model.predict_proba(X_test)[:,1]
y_pred0 = (y_proba >= 0.5).astype(int)

print("ROC-AUC:", roc_auc_score(y_test, y_proba).round(3))
print("\nClassification Report @0.5:")
print(classification_report(y_test, y_pred0, target_names=['No-Show', 'Show']))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred0))

# %% [markdown]
# ## 5. Threshold Tuning for Target Recall

# %% [code]
prec, rec, th = precision_recall_curve(y_test, y_proba)

# Choose threshold where recall on No-Show ≥ 0.50

```

```

target_recall = 0.50
idx = np.argmax(rec >= target_recall)
best_thresh = th[idx]
print(f"\nThreshold for ≥{target_recall*100:.0f}% recall: {best_thresh:.2f}")
print(f"Precision at that recall: {prec[idx]:.2f}, Recall: {rec[idx]:.2f}")

# Recompute predictions
y_pred_tuned = (y_proba >= best_thresh).astype(int)

print("\nClassification Report @tuned threshold:")
print(classification_report(y_test, y_pred_tuned, target_names=['No-Show', 'Show']))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tuned))

# %% [markdown]
# ## 6. Visualizations

# %% [code]
# ROC Curve
RocCurveDisplay.from_predictions(y_test, y_proba)
plt.title("ROC Curve")
plt.show()

# Precision-Recall Curve
PrecisionRecallDisplay.from_predictions(y_test, y_proba)
plt.title("Precision-Recall Curve")
plt.show()

# %% [markdown]
# ## 7. Save Model & Threshold

# %% [code]
joblib.dump(model, 'no_show_model_tuned.pkl')
with open('best_threshold.txt', 'w') as f:
    f.write(str(best_thresh))
print("Saved tuned model and threshold.")

```

C:\Users\palak\AppData\Roaming\Python\Python312\site-packages\xgboost\training.py:183: UserWarning: [22:28:14] WARNING: C:\actions-runner_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

bst.update(dtrain, iteration=i, fobj=obj)

ROC-AUC: 0.736

Classification Report @0.5:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| No-Show | 0.32 | 0.76 | 0.45 | 4336 |
| Show | 0.90 | 0.58 | 0.71 | 17062 |
| accuracy | | | 0.62 | 21398 |
| macro avg | 0.61 | 0.67 | 0.58 | 21398 |
| weighted avg | 0.78 | 0.62 | 0.66 | 21398 |

Confusion Matrix:

```
[[3275 1061]
 [7088 9974]]
```

Threshold for ≥50% recall: 0.04

Precision at that recall: 0.80, Recall: 1.00

Classification Report @tuned threshold:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| No-Show | 0.00 | 0.00 | 0.00 | 4336 |
| Show | 0.80 | 1.00 | 0.89 | 17062 |
| accuracy | | | 0.80 | 21398 |
| macro avg | 0.40 | 0.50 | 0.44 | 21398 |
| weighted avg | 0.64 | 0.80 | 0.71 | 21398 |

Confusion Matrix:

```
[[ 0 4336]
 [ 0 17062]]
```

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

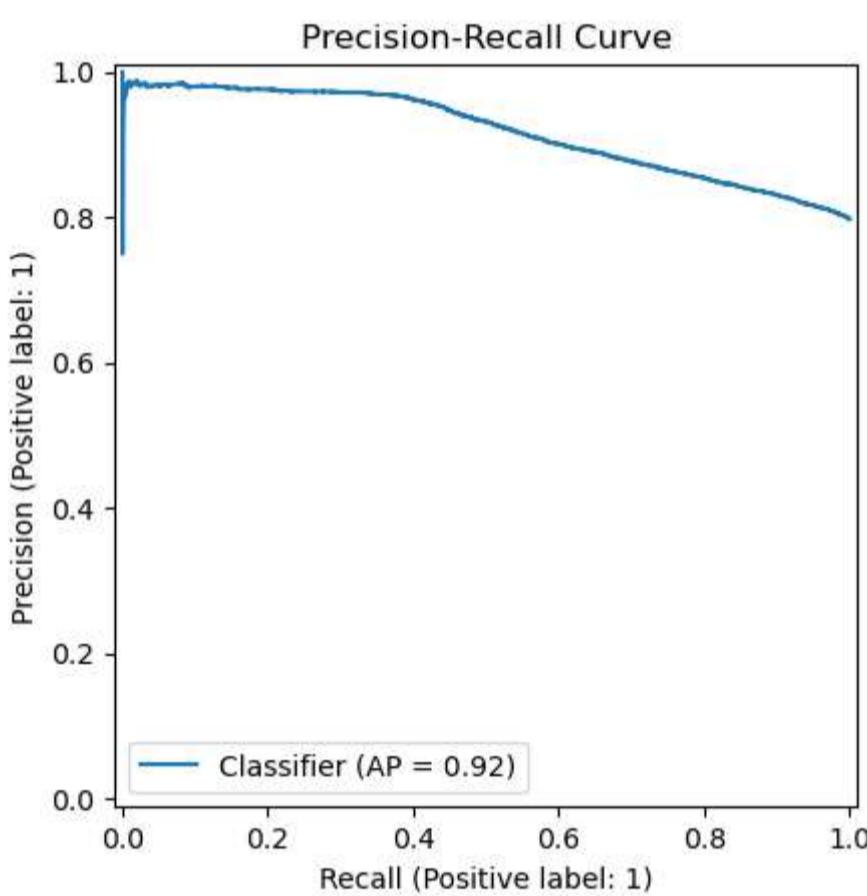
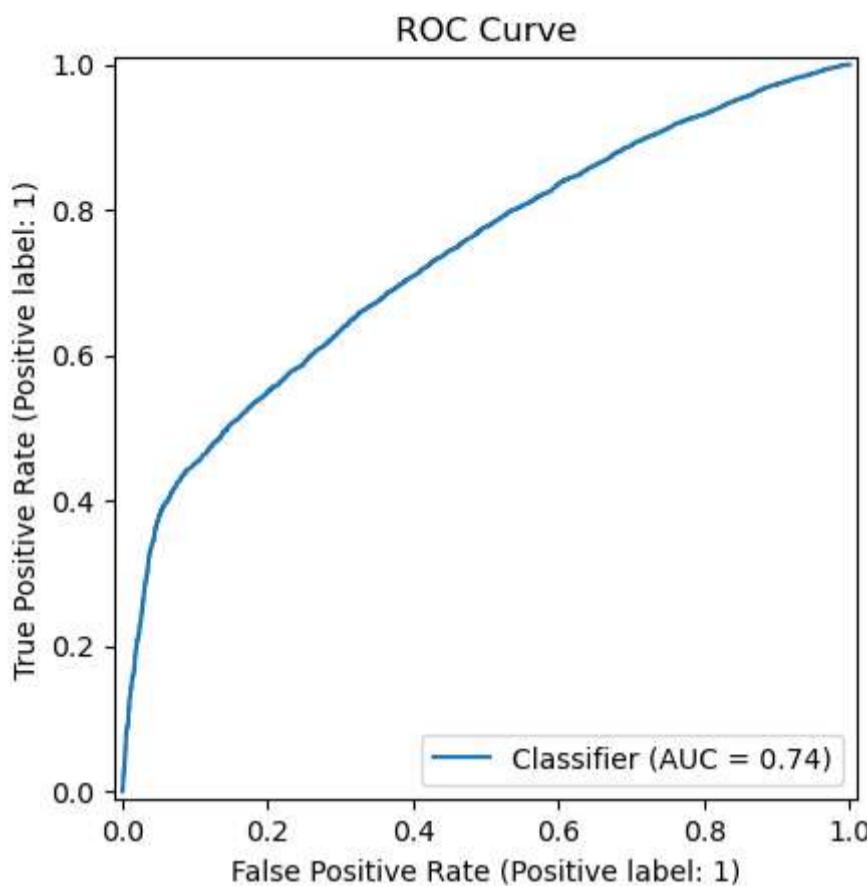
```
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



Saved tuned model and threshold.

```
In [34]: # version3:
import pandas as pd
import numpy as np
import joblib
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    roc_auc_score, confusion_matrix, classification_report,
    precision_recall_curve, PrecisionRecallDisplay
)

# 1. Reload your cleaned data & model
df      = pd.read_csv('healthcare_noshow_appt.csv',
                      parse_dates=['ScheduledDay', 'AppointmentDay'],
                      na_values=['NaN'])
model = joblib.load('no_show_model.pkl')

# Rebuild X, y exactly as before...
# (Assume you have your build_features function from above)
X, y = build_features(df)

# 2. Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# 3. Get "No-Show" probability: class 0
y_proba_ns = model.predict_proba(X_test)[:,0]
y_true_ns  = (y_test == 0).astype(int)      # 1 if No-Show, 0 otherwise

# 4. ROC-AUC for No-Show detection
auc_ns = roc_auc_score(y_true_ns, y_proba_ns)
print("ROC-AUC (No-Show):", auc_ns.round(3))
```

```

# 5. Precision-Recall tuning
prec, rec, thresh = precision_recall_curve(y_true_ns, y_proba_ns)

# For example, pick threshold for ≥50% No-Show recall
target_recall = 0.50
idx = np.argmax(rec >= target_recall)
best_thresh_ns = thresh[idx]
print(f"Threshold for ≥{target_recall*100:.0f}% No-Show recall: {best_thresh_ns:.3f}")
print(f"Precision at that recall: {prec[idx]:.3f}, Recall: {rec[idx]:.3f}")

# 6. Evaluate at that threshold
y_pred_ns = (y_proba_ns >= best_thresh_ns).astype(int)
print("\nClassification Report (No-Show positive):")
print(classification_report(y_true_ns, y_pred_ns, target_names=['Show', 'No-Show']))
print("Confusion Matrix:\n", confusion_matrix(y_true_ns, y_pred_ns))

# 7. Plot Precision-Recall curve
import matplotlib.pyplot as plt
PrecisionRecallDisplay.from_predictions(y_true_ns, y_proba_ns)
plt.title("Precision-Recall Curve for No-Show Detection")
plt.show()

```

ROC-AUC (No-Show): 0.738

Threshold for ≥50% No-Show recall: 0.000

Precision at that recall: 0.203, Recall: 1.000

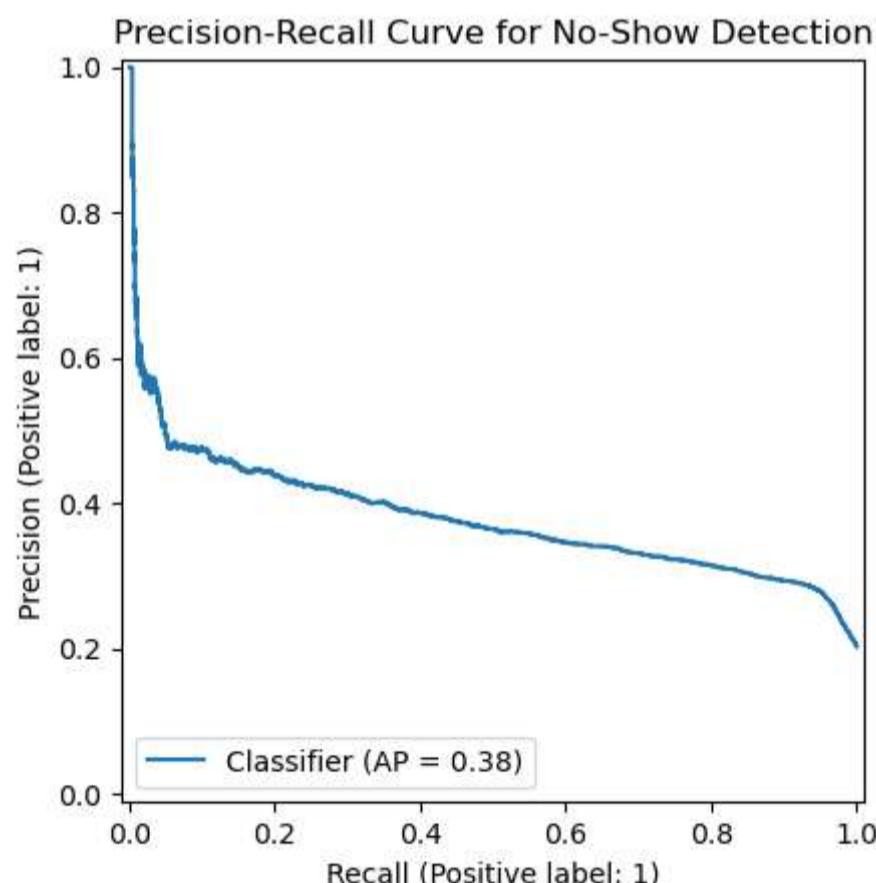
Classification Report (No-Show positive):

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| Show | 0.00 | 0.00 | 0.00 | 17062 |
| No-Show | 0.20 | 1.00 | 0.34 | 4336 |
| accuracy | | | 0.20 | 21398 |
| macro avg | 0.10 | 0.50 | 0.17 | 21398 |
| weighted avg | 0.04 | 0.20 | 0.07 | 21398 |

Confusion Matrix:

```
[[ 0 17062]
 [ 0 4336]]
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



Right now tuned for 100% recall, which pushed the threshold down to essentially 0.0, so your model predicts "No-Show" for every single appointment. That gets you perfect recall but zero precision (and 20% accuracy, since ~20% of cases are true no-shows).

- A more practical approach is to pick the threshold that maximizes F₁-score (the balance of precision and recall) for the No-Show class.
Here's how to do that:

```
In [36]: import numpy as np
from sklearn.metrics import f1_score

# y_true_ns and y_proba_ns from before
# y_true_ns = (y_test == 0).astype(int)
# y_proba_ns = model.predict_proba(X_test)[:,0]

# 1. Define a grid of candidate thresholds (exclude 0 and 1)
thresholds = np.linspace(0.01, 0.99, 99)

best_f1 = 0
best_thresh = 0.5

for t in thresholds:
    y_pred = (y_proba_ns >= t).astype(int)
    f1 = f1_score(y_true_ns, y_pred)
    if f1 > best_f1:
        best_f1 = f1
        best_thresh = t

print(f"Best F1={best_f1:.3f} at threshold={best_thresh:.2f}")

# 2. Evaluate at that threshold
y_pred_best = (y_proba_ns >= best_thresh).astype(int)
print("\nClassification Report (No-Show positive):")
print(classification_report(y_true_ns, y_pred_best, target_names=['Show', 'No-Show']))
print("Confusion Matrix:\n", confusion_matrix(y_true_ns, y_pred_best))
```

Best F1=0.452 at threshold=0.21

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| Show | 0.91 | 0.59 | 0.72 | 17062 |
| No-Show | 0.32 | 0.76 | 0.45 | 4336 |
| accuracy | | | 0.63 | 21398 |
| macro avg | 0.61 | 0.68 | 0.59 | 21398 |
| weighted avg | 0.79 | 0.63 | 0.66 | 21398 |

Confusion Matrix:
[[10141 6921]
[1048 3288]]

0.21 cutoff gives you the best balance (F₁≈0.45), which means:

- You'll flag ~76% of true no-shows (recall)
- About one-third of your flags will actually be no-shows (precision)

to run streamlit :

- cd C:\Users\palak\Desktop\EMR.Dashboard
- streamlit run streamlit_app.py

Advanced Analysis:

```
In [38]: !pip install shap
```

```

Defaulting to user installation because normal site-packages is not writeable
Collecting shap
  Downloading shap-0.47.2-cp312-cp312-win_amd64.whl.metadata (25 kB)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from shap) (1.26.4)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from shap) (1.13.1)
Requirement already satisfied: scikit-learn in c:\programdata\anaconda3\lib\site-packages (from shap) (1.5.1)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in c:\programdata\anaconda3\lib\site-packages (from shap) (4.66.5)
Requirement already satisfied: packaging>20.9 in c:\programdata\anaconda3\lib\site-packages (from shap) (24.1)
Collecting slicer==0.0.8 (from shap)
  Downloading slicer-0.0.8-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: numba>=0.54 in c:\programdata\anaconda3\lib\site-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in c:\programdata\anaconda3\lib\site-packages (from shap) (3.0.0)
Requirement already satisfied: typing-extensions in c:\programdata\anaconda3\lib\site-packages (from shap) (4.11.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in c:\programdata\anaconda3\lib\site-packages (from numba>=0.54->shap) (0.43.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\programdata\anaconda3\lib\site-packages (from pandas->shap) (2.9.0-post0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas->shap) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\programdata\anaconda3\lib\site-packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.2.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn->shap) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
Downloading shap-0.47.2-cp312-cp312-win_amd64.whl (545 kB)
----- 0.0/545.2 kB ? eta ------
----- 262.1/545.2 kB ? eta ------
----- 524.3/545.2 kB 1.5 MB/s eta 0:00:01
----- 545.2/545.2 kB 670.8 kB/s eta 0:00:00
Downloading slicer-0.0.8-py3-none-any.whl (15 kB)
Installing collected packages: slicer, shap
Successfully installed shap-0.47.2 slicer-0.0.8

```

```

In [40]: import shap
import matplotlib.pyplot as plt

# ... assuming df, model, build_features, X, y are already defined ...

# Sample for speed
X_sample = X.sample(2000, random_state=42)

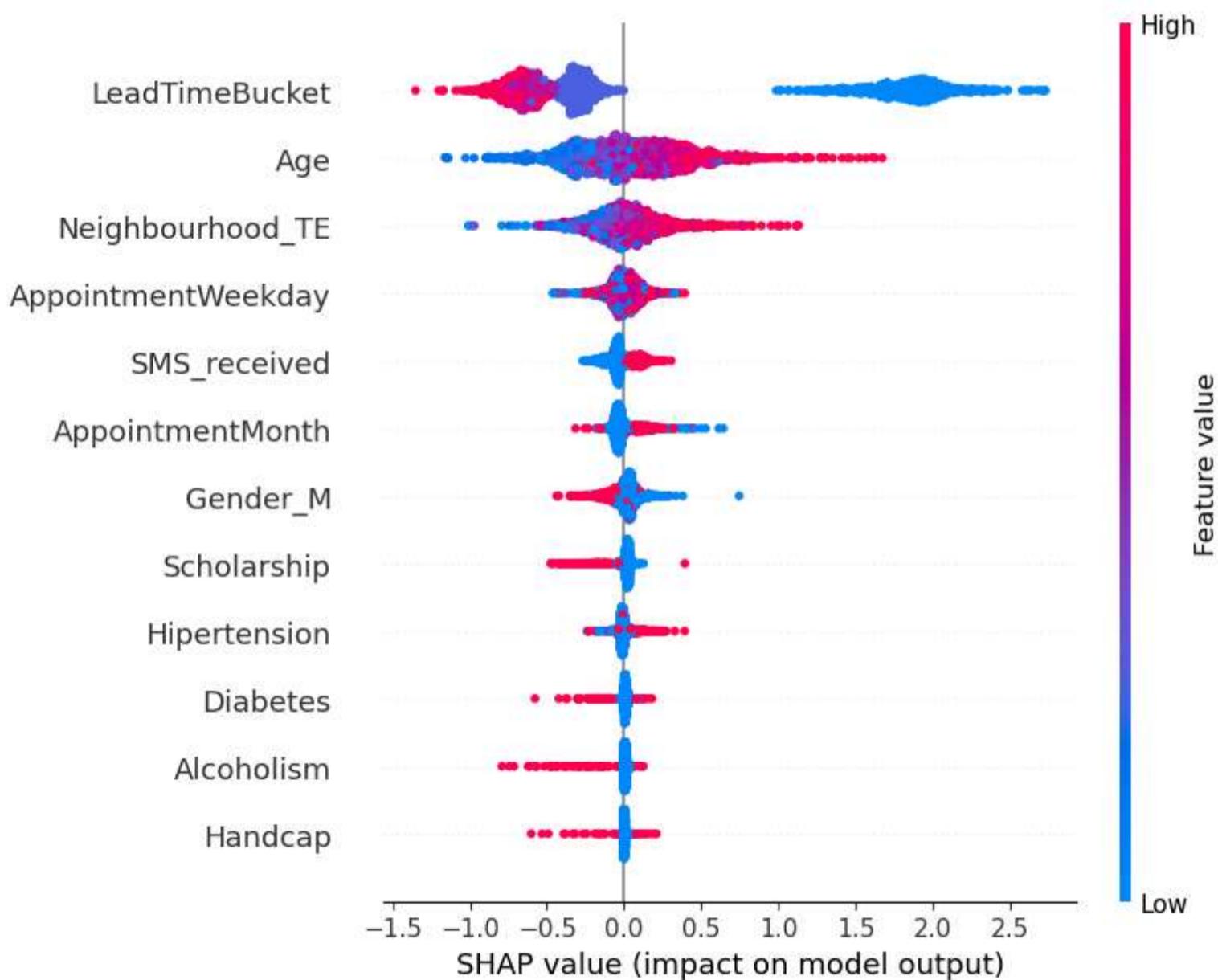
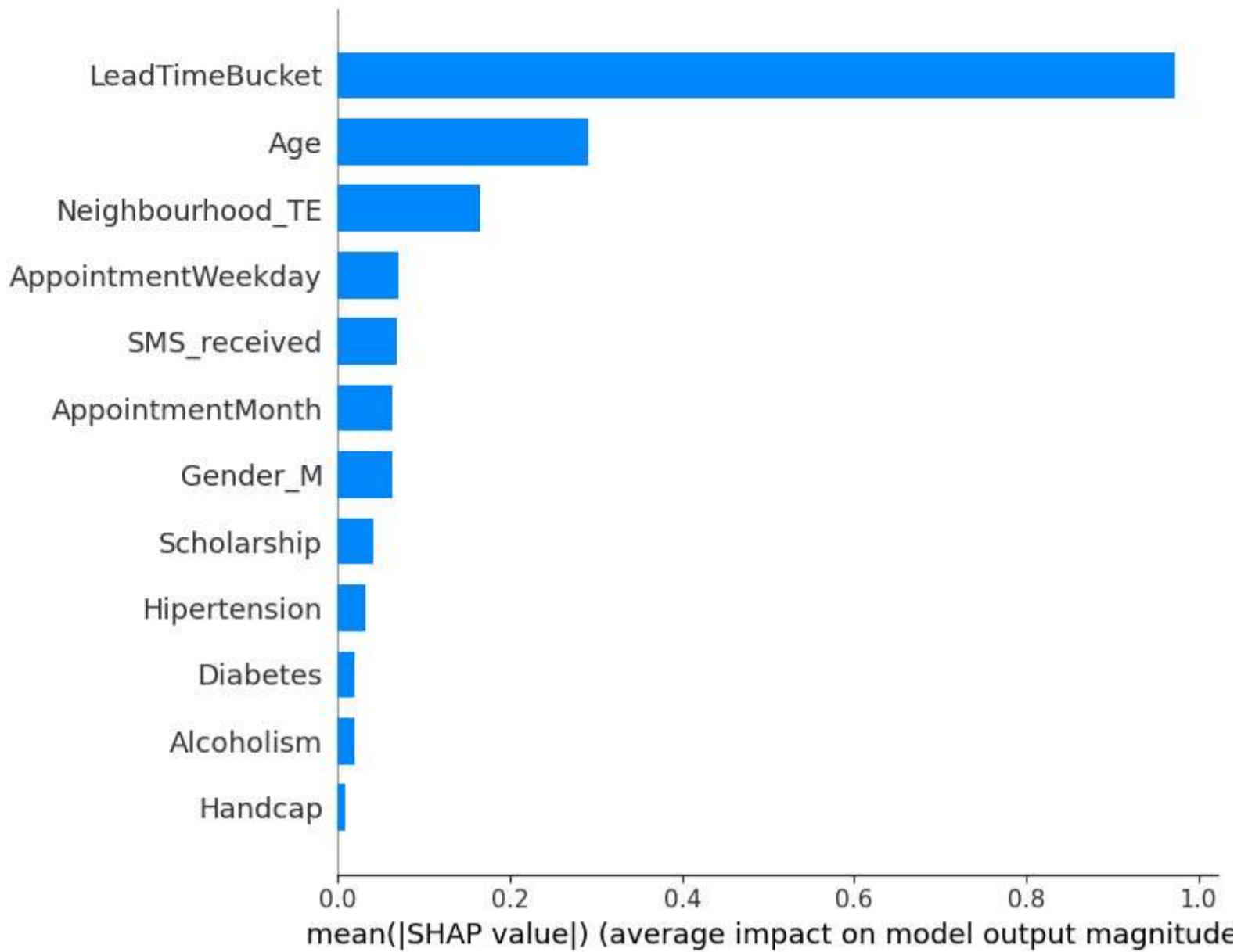
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_sample)

# Global bar plot
plt.figure(figsize=(8,6))
shap.summary_plot(shap_values, X_sample, plot_type='bar', show=False)
plt.tight_layout()
plt.show()

# Beeswarm
plt.figure(figsize=(10,8))
shap.summary_plot(shap_values, X_sample, show=False)
plt.tight_layout()
plt.show()

# Force plots for first and last by position
for pos in [0, -1]:
    row = X_sample.iloc[pos]
    sv = shap_values[pos]
    print(f"\nSHAP Force Plot for sample position {pos} (orig idx {row.name}):")
    shap.force_plot(
        explainer.expected_value,
        sv,
        row,
        matplotlib=True
    )

```



SHAP Force Plot for sample position 0 (orig idx 87067):



SHAP Force Plot for sample position -1 (orig idx 55927):



Continued in PowerBI :