Load Data and Import Dependencies

In [2]:
```
!pip install mglearn
```

```
Requirement already satisfied: mglearn in c:\users\administrator\anaconda3\lib\site-
packages (0.2.0)
Requirement already satisfied: numpy in c:\users\administrator\anaconda3\lib\site-pa
ckages (from mglearn) (1.26.4)
Requirement already satisfied: matplotlib in c:\users\administrator\anaconda3\lib\si
te-packages (from mglearn) (3.9.2)
Requirement already satisfied: scikit-learn in c:\users\administrator\anaconda3\lib
\site-packages (from mglearn) (1.5.1)
Requirement already satisfied: pandas in c:\users\administrator\anaconda3\lib\site-p
ackages (from mglearn) (2.2.2)
Requirement already satisfied: pillow in c:\users\administrator\anaconda3\lib\site-p
ackages (from mglearn) (10.4.0)
Requirement already satisfied: cycler in c:\users\administrator\anaconda3\lib\site-p
ackages (from mglearn) (0.11.0)
Requirement already satisfied: imageio in c:\users\administrator\anaconda3\lib\site-
packages (from mglearn) (2.33.1)
Requirement already satisfied: joblib in c:\users\administrator\anaconda3\lib\site-p
ackages (from mglearn) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\administrator\anaconda3
\lib\site-packages (from matplotlib->mglearn) (1.2.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\administrator\anaconda3
\lib\site-packages (from matplotlib->mglearn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\administrator\anaconda3
\lib\site-packages (from matplotlib->mglearn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\administrator\anaconda3\l
ib\site-packages (from matplotlib->mglearn) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\administrator\anaconda3
\lib\site-packages (from matplotlib->mglearn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\administrator\anacon
da3\lib\site-packages (from matplotlib->mglearn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\administrator\anaconda3\lib
\site-packages (from pandas->mglearn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\administrator\anaconda3\li
b\site-packages (from pandas->mglearn) (2023.3)
Requirement already satisfied: scipy>=1.6.0 in c:\users\administrator\anaconda3\lib
\site-packages (from scikit-learn->mglearn) (1.13.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\administrator\anacon
da3\lib\site-packages (from scikit-learn->mglearn) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\users\administrator\anaconda3\lib\site
-packages (from python-dateutil>=2.7->matplotlib->mglearn) (1.16.0)
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
```

```python
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
```

In [4]:
```python
df = pd.read_csv('data/KaggleV2-May-2016.csv')
df
```

Out[4]:

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

110527 rows × 14 columns

Analysis and Visualization

In [6]:
```python
df.shape
```

Out[6]: (110527, 14)

In [7]:
```python
df.describe()
```

Out[7]:

	PatientId	AppointmentID	Age	Scholarship	Hipertension	D
count	1.105270e+05	1.105270e+05	110527.000000	110527.000000	110527.000000	110527
mean	1.474963e+14	5.675305e+06	37.088874	0.098266	0.197246	0
std	2.560949e+14	7.129575e+04	23.110205	0.297675	0.397921	0
min	3.921784e+04	5.030230e+06	-1.000000	0.000000	0.000000	0
25%	4.172614e+12	5.640286e+06	18.000000	0.000000	0.000000	0
50%	3.173184e+13	5.680573e+06	37.000000	0.000000	0.000000	0
75%	9.439172e+13	5.725524e+06	55.000000	0.000000	0.000000	0
max	9.999816e+14	5.790484e+06	115.000000	1.000000	1.000000	1

In [8]:

```
df.head
```

Out[8]:

```
<bound method NDFrame.head of             PatientId  AppointmentID Gender
ScheduledDay  \
0       2.987250e+13        5642903      F  2016-04-29T18:38:08Z
1       5.589978e+14        5642503      M  2016-04-29T16:08:27Z
2       4.262962e+12        5642549      F  2016-04-29T16:19:04Z
3       8.679512e+11        5642828      F  2016-04-29T17:29:31Z
4       8.841186e+12        5642494      F  2016-04-29T16:07:23Z
...              ...            ...    ...                   ...
110522  2.572134e+12        5651768      F  2016-05-03T09:15:35Z
110523  3.596266e+12        5650093      F  2016-05-03T07:27:33Z
110524  1.557663e+13        5630692      F  2016-04-27T16:03:52Z
110525  9.213493e+13        5630323      F  2016-04-27T15:09:23Z
110526  3.775115e+14        5629448      F  2016-04-27T13:30:56Z

              AppointmentDay  Age      Neighbourhood  Scholarship  \
0       2016-04-29T00:00:00Z   62    JARDIM DA PENHA            0
1       2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0
2       2016-04-29T00:00:00Z   62      MATA DA PRAIA            0
3       2016-04-29T00:00:00Z    8  PONTAL DE CAMBURI            0
4       2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0
...                      ...  ...                ...          ...
110522  2016-06-07T00:00:00Z   56        MARIA ORTIZ            0
110523  2016-06-07T00:00:00Z   51        MARIA ORTIZ            0
110524  2016-06-07T00:00:00Z   21        MARIA ORTIZ            0
110525  2016-06-07T00:00:00Z   38        MARIA ORTIZ            0
110526  2016-06-07T00:00:00Z   54        MARIA ORTIZ            0

        Hipertension  Diabetes  Alcoholism  Handcap  SMS_received No-show
0                  1         0           0        0             0      No
1                  0         0           0        0             0      No
2                  0         0           0        0             0      No
3                  0         0           0        0             0      No
4                  1         1           0        0             0      No
...              ...       ...         ...      ...           ...     ...
110522             0         0           0        0             1      No
110523             0         0           0        0             1      No
110524             0         0           0        0             1      No
110525             0         0           0        0             1      No
110526             0         0           0        0             1      No

[110527 rows x 14 columns]>
```

In [9]:

```
print(df.isnull().sum())
```

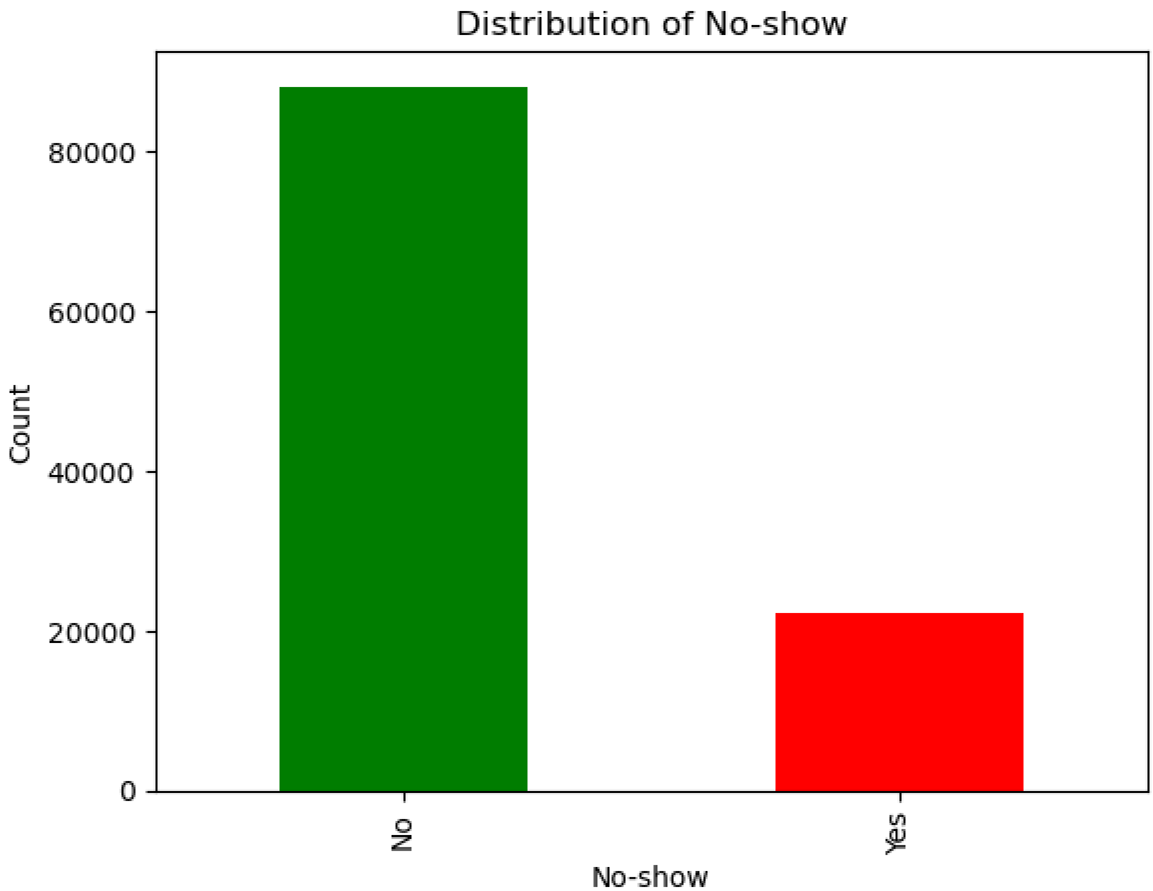
```
PatientId         0
AppointmentID     0
Gender            0
ScheduledDay      0
AppointmentDay    0
Age               0
Neighbourhood     0
Scholarship       0
Hipertension      0
Diabetes          0
Alcoholism        0
Handcap           0
SMS_received      0
No-show           0
dtype: int64
```

In [10]:
```
df.dtypes
```

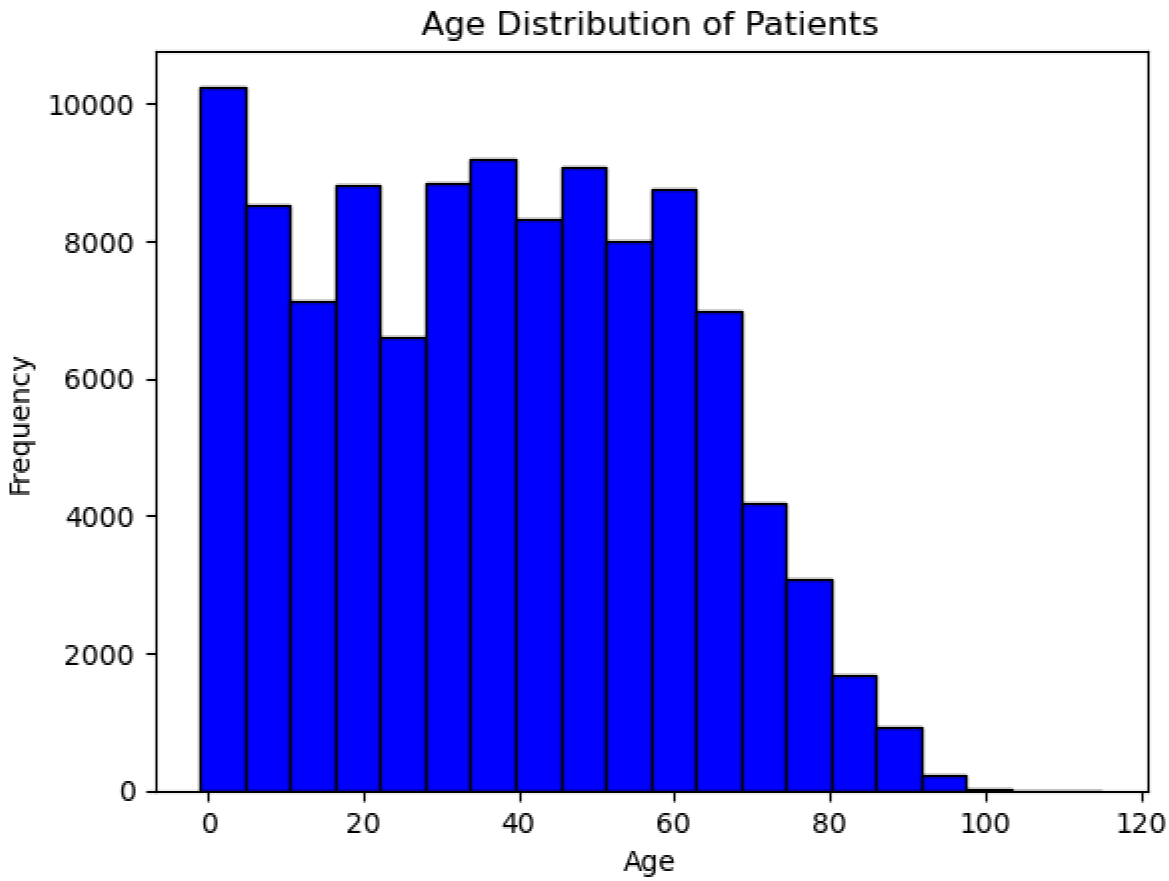
Out[10]:
```
PatientId         float64
AppointmentID       int64
Gender             object
ScheduledDay       object
AppointmentDay     object
Age                 int64
Neighbourhood      object
Scholarship         int64
Hipertension        int64
Diabetes            int64
Alcoholism          int64
Handcap             int64
SMS_received        int64
No-show            object
dtype: object
```

In [11]:
```
# df['Gender'] = pd.to_numeric(df['Gender'], downcast='integer', errors='coerce')
# df.dtypes
```

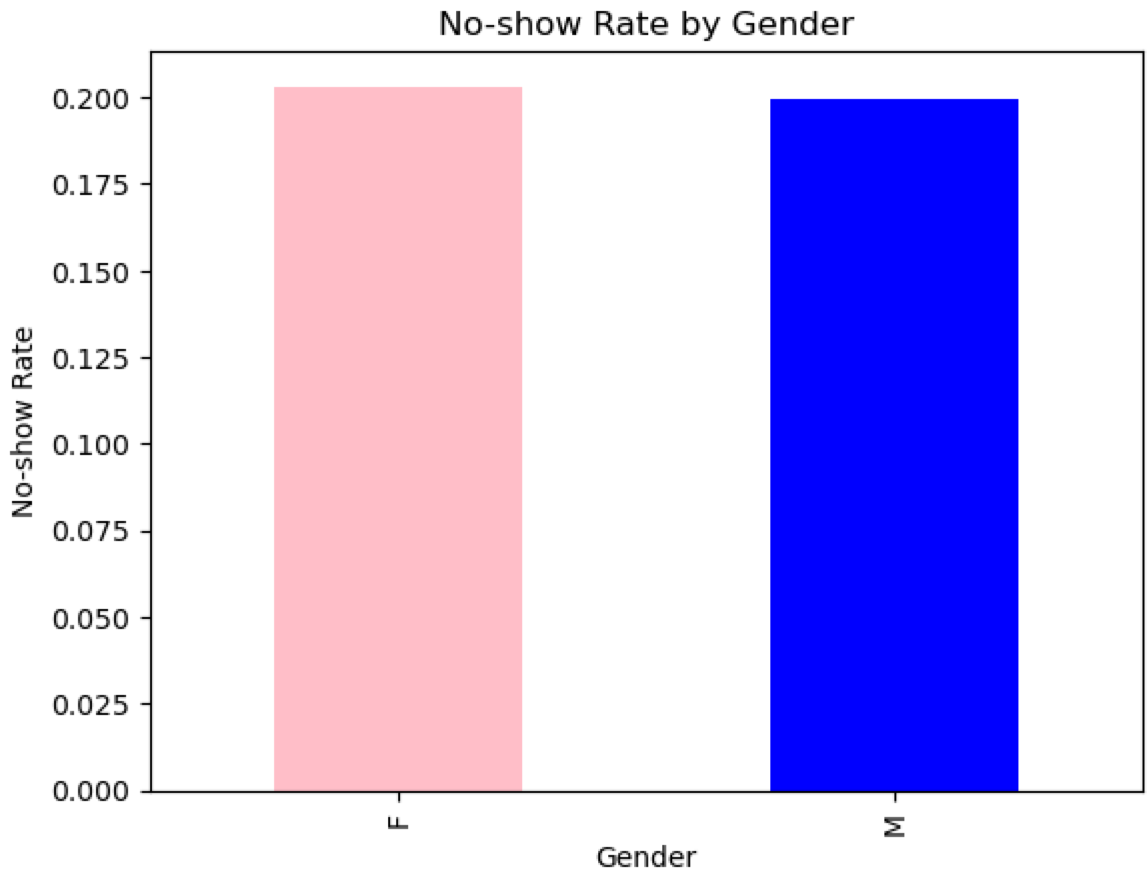
In [12]:
```
# Plotting the distribution of No-show
df['No-show'].value_counts().plot(kind='bar', color=['green', 'red'])
plt.title('Distribution of No-show')
plt.xlabel('No-show')
plt.ylabel('Count')
plt.show()
```



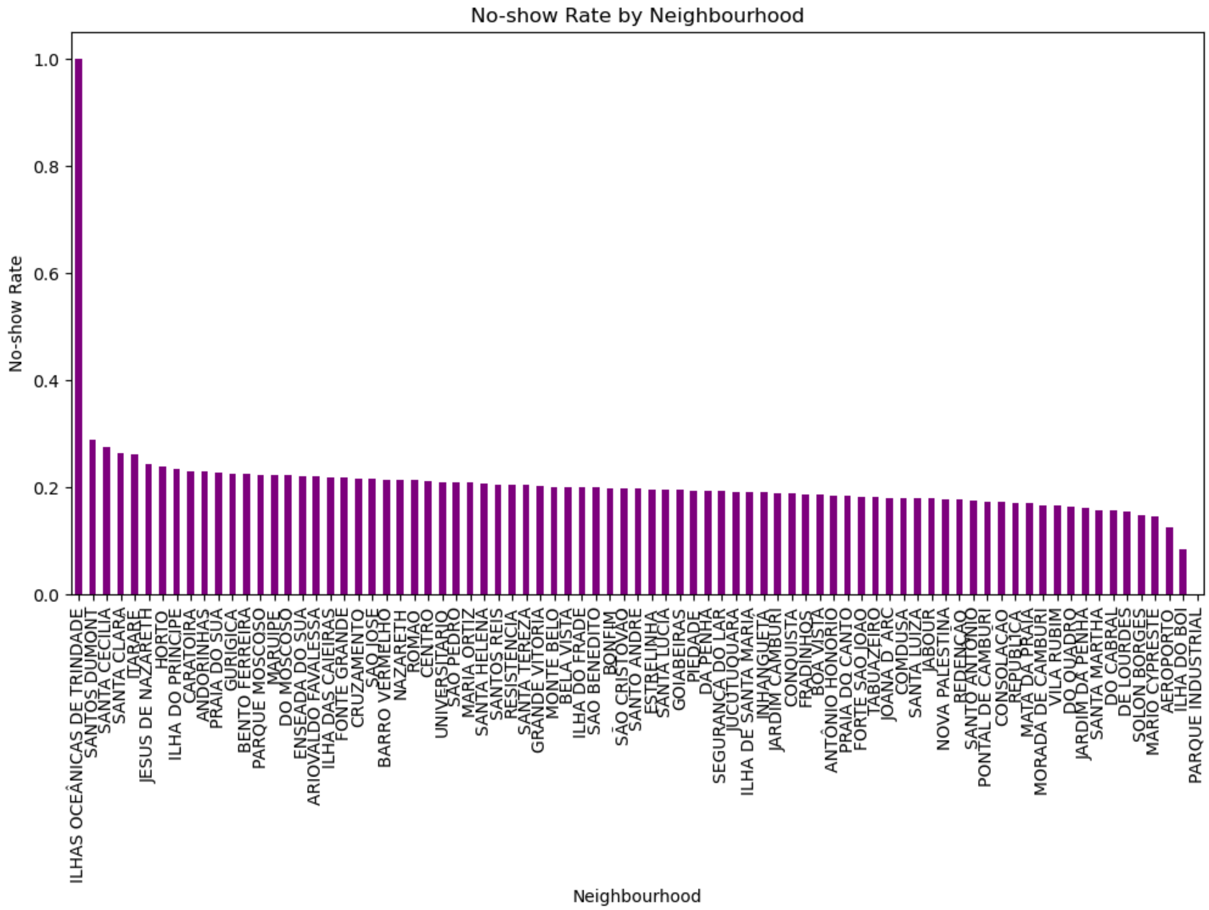
In [13]:
```
# Plotting the age distribution
plt.hist(df['Age'], bins=20, color='blue', edgecolor='black')
plt.title('Age Distribution of Patients')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



In [14]:
```
# Calculating no-show rate by gender
no_show_rate = df.groupby('Gender')['No-show'].apply(lambda x: (x == 'Yes').mean())

# Plotting the no-show rate by gender
no_show_rate.plot(kind='bar', color=['pink', 'blue'])
plt.title('No-show Rate by Gender')
plt.xlabel('Gender')
plt.ylabel('No-show Rate')
plt.show()
```

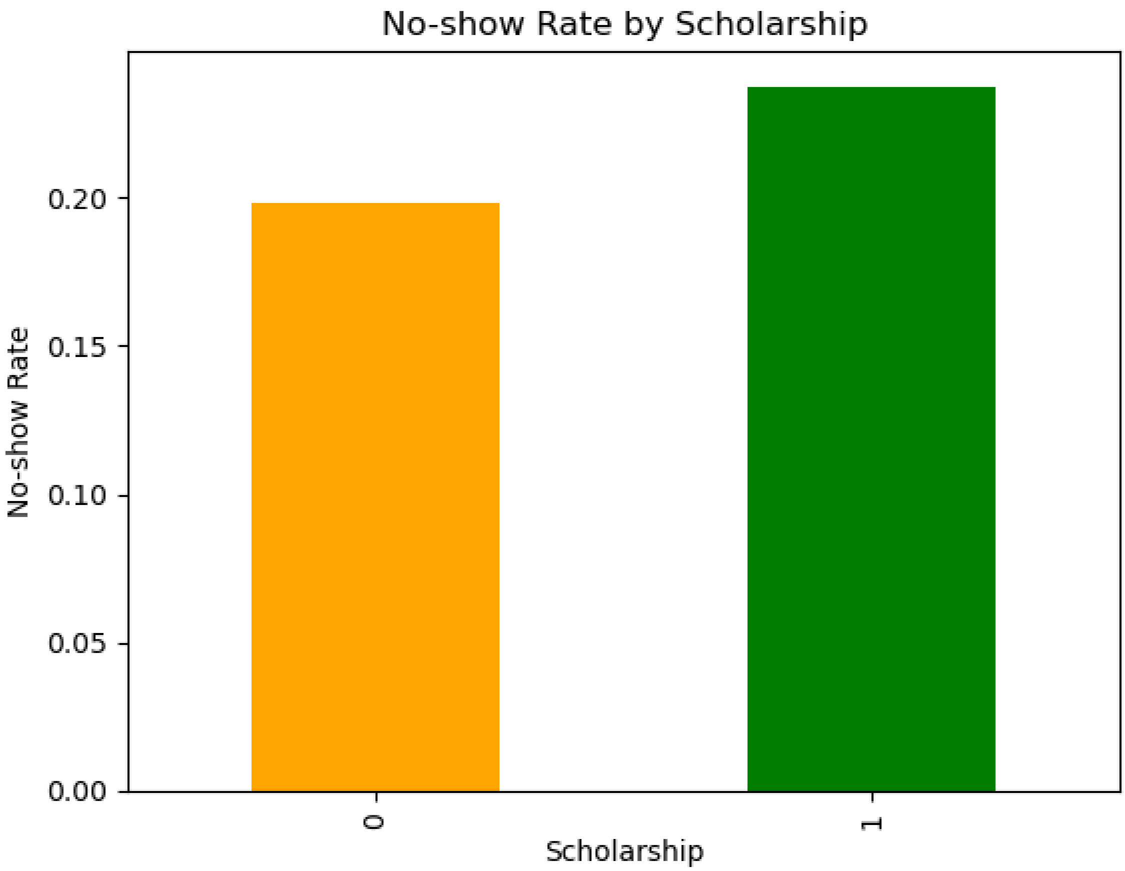


In [15]:
```
# Calculating no-show rate by neighbourhood
no_show_rate_neighbourhood = df.groupby('Neighbourhood')['No-show'].apply(lambda x:

# Plotting the no-show rate by neighbourhood
no_show_rate_neighbourhood.sort_values(ascending=False).plot(kind='bar', figsize=(1
plt.title('No-show Rate by Neighbourhood')
plt.xlabel('Neighbourhood')
plt.ylabel('No-show Rate')
plt.show()
```

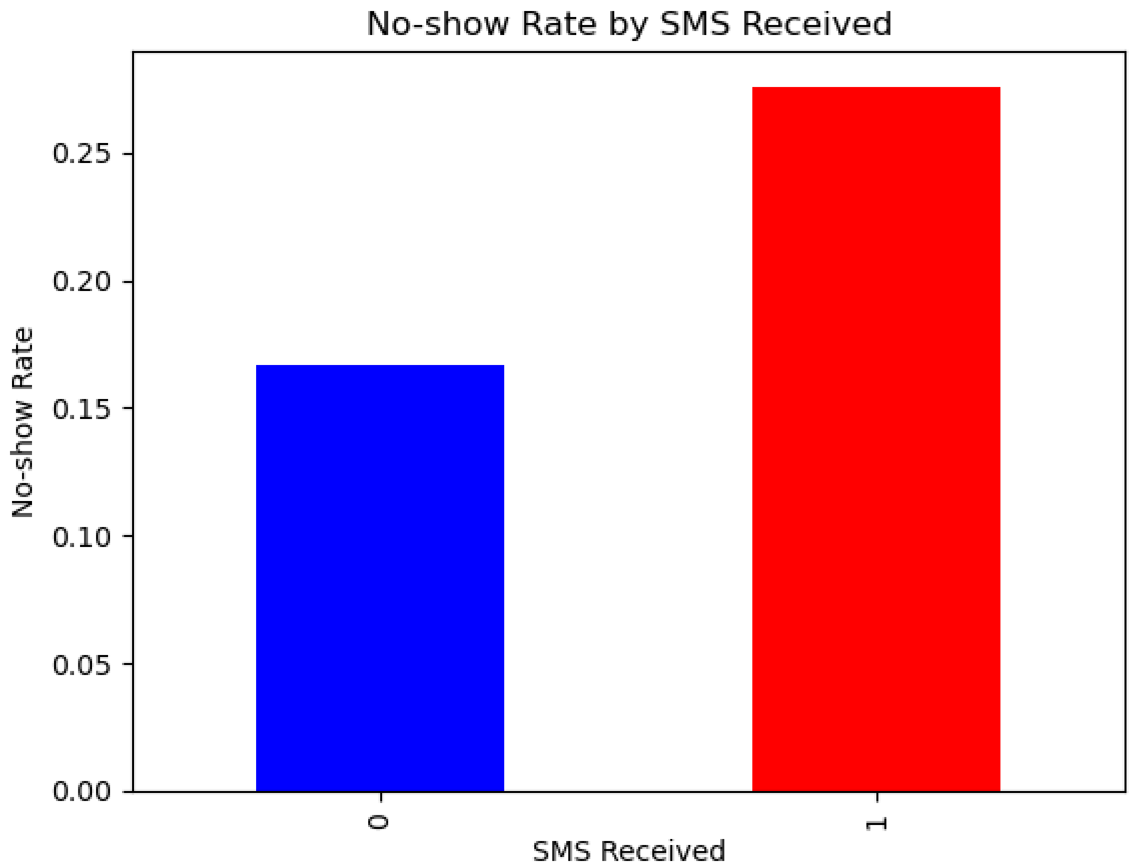


In [16]:

```
# Calculating no-show rate by scholarship
no_show_rate_scholarship = df.groupby('Scholarship')['No-show'].apply(lambda x: (x

# Plotting the no-show rate by scholarship
no_show_rate_scholarship.plot(kind='bar', color=['orange', 'green'])
plt.title('No-show Rate by Scholarship')
plt.xlabel('Scholarship')
plt.ylabel('No-show Rate')
plt.show()
```

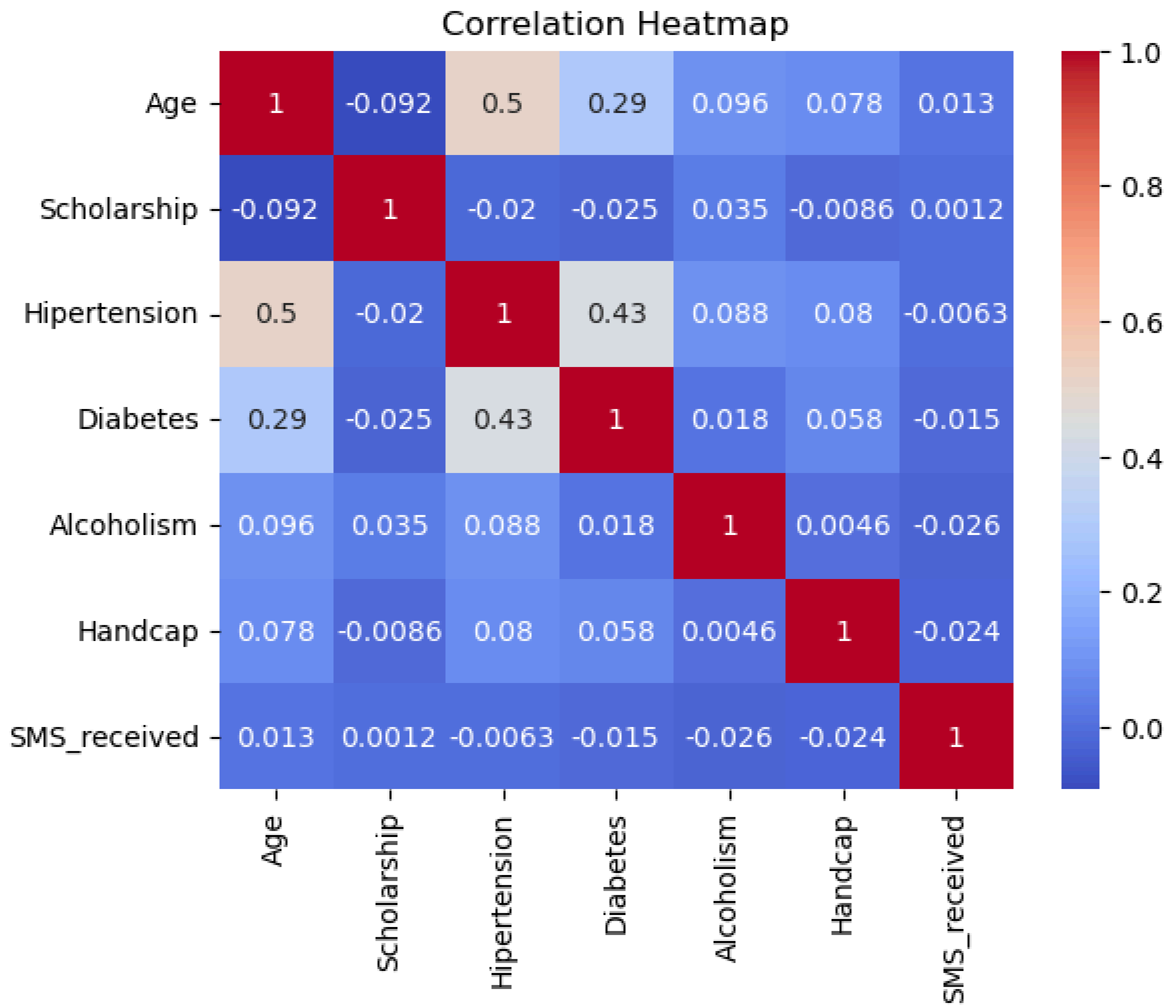


In [17]:
```
# Calculating no-show rate by SMS received
no_show_rate_sms = df.groupby('SMS_received')['No-show'].apply(lambda x: (x == 'Yes

# Plotting the no-show rate by SMS received
no_show_rate_sms.plot(kind='bar', color=['blue', 'red'])
plt.title('No-show Rate by SMS Received')
plt.xlabel('SMS Received')
plt.ylabel('No-show Rate')
plt.show()
```



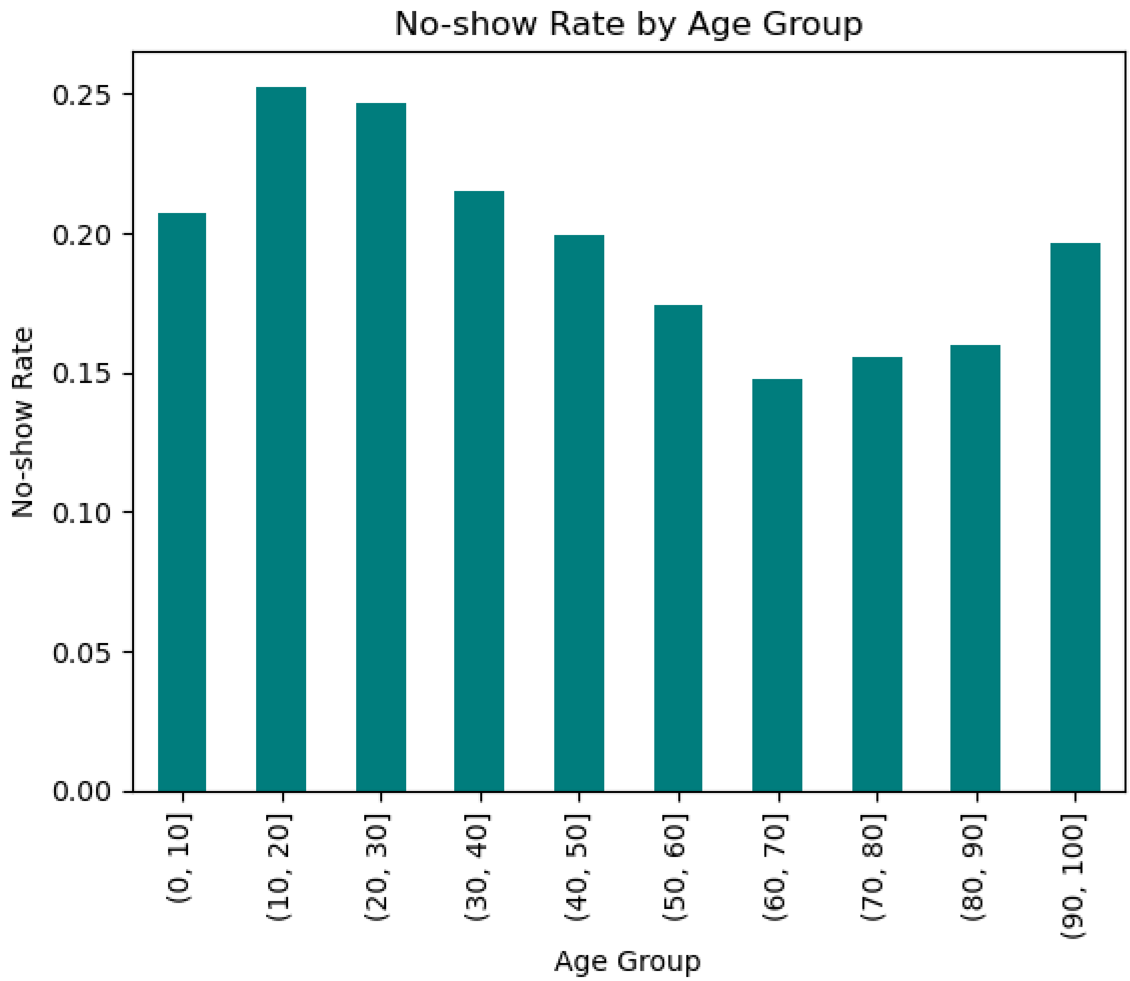
In [18]:

```
# Selecting numerical columns for correlation
numerical_columns = ['Age', 'Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism'

# Calculating the correlation matrix
corr_matrix = df[numerical_columns].corr()

# Plotting the heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [19]:
```python
# Creating age groups
df['AgeGroup'] = pd.cut(df['Age'], bins=range(0, 101, 10))

# Calculating no-show rate by age group
no_show_rate_age = df.groupby('AgeGroup')['No-show'].apply(lambda x: (x == 'Yes').m

# Plotting the no-show rate by age group
no_show_rate_age.plot(kind='bar', color='teal')
plt.title('No-show Rate by Age Group')
plt.xlabel('Age Group')
plt.ylabel('No-show Rate')
plt.show()
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_11032\2774455716.py:5: FutureWar
ning: The default of observed=False is deprecated and will be changed to True in a f
uture version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.
  no_show_rate_age = df.groupby('AgeGroup')['No-show'].apply(lambda x: (x == 'Yes').
mean())
```



Feature Engineering - Selection, Cleaning

Cleaning

In [22]:
```
# Check for missing values
print(df.isnull().sum())

# Remove duplicates
df = df.drop_duplicates()
```

```
PatientId              0
AppointmentID          0
Gender                 0
ScheduledDay           0
AppointmentDay         0
Age                    0
Neighbourhood          0
Scholarship            0
Hipertension           0
Diabetes               0
Alcoholism             0
Handcap                0
SMS_received           0
No-show                0
AgeGroup            3547
dtype: int64
```

In [23]:

```
# Convert ScheduledDay and AppointmentDay to datetime
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])

# Check unique values in categorical columns
print(df['Gender'].unique())
print(df['No-show'].unique())
```

```
['F' 'M']
['No' 'Yes']
```

Create New Features

In [25]:

```
# Calculate waiting time in days
df['WaitingTime'] = (df['AppointmentDay'] - df['ScheduledDay']).dt.days

# Extract day of the week from AppointmentDay
df['AppointmentDayOfWeek'] = df['AppointmentDay'].dt.day_name()

# Bin Age into groups
df['AgeGroup'] = pd.cut(df['Age'], bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

# One-hot encode categorical variables
df = pd.get_dummies(df, columns=['Gender', 'Neighbourhood', 'AppointmentDayOfWeek',


# Encode No-show column
df['No-show'] = df['No-show'].apply(lambda x: 1 if x == 'Yes' else 0)

# Drop unnecessary columns
df = df.drop(['PatientId', 'AppointmentID', 'ScheduledDay', 'AppointmentDay'], axis
```

In [26]:

```
df.shape
```

Out[26]:

```
(110527, 104)
```

Data Splitting

In [28]:
```
from sklearn.model_selection import train_test_split

# Define features (X) and target (y)
X = df.drop('No-show', axis=1)
y = df['No-show']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [29]:
```
from sklearn.preprocessing import StandardScaler

# Initialize the scaler
scaler = StandardScaler()

# Scale numerical features
X_train[['Age', 'WaitingTime']] = scaler.fit_transform(X_train[['Age', 'WaitingTime
X_test[['Age', 'WaitingTime']] = scaler.transform(X_test[['Age', 'WaitingTime']])
```

In [30]:
```
X_train.shape
```

Out[30]:
```
(88421, 103)
```

In [31]:
```
y_train.shape
```

Out[31]:
```
(88421,)
```

Model selection - best fit analysis

In [33]:
```
logistic = LogisticRegression(max_iter=100000)
logistic.fit(X,y)

print("Accuracy score on training {:.5f}".format(logistic.score(X_train,y_train)))
print("Accuracy score on testing {:.5f}".format(logistic.score(X_test,y_test)))
```

```
Accuracy score on training 0.79776
Accuracy score on testing 0.79929
```

In [34]:
```
logistic100 = LogisticRegression(C=10000000, max_iter=100000)
logistic100.fit(X,y)

print("Accuracy score on training {:.5f}".format(logistic100.score(X_train,y_train)
print("Accuracy score on testing {:.5f}".format(logistic100.score(X_test,y_test)))
```

```
Accuracy score on training 0.79776
Accuracy score on testing 0.79929
```

In [35]:
```
logistic001 = LogisticRegression(C=0.001, max_iter=100000)
logistic001.fit(X,y)
```

```
print("Accuracy score on training {:.5f}".format(logistic001.score(X_train,y_train)
print("Accuracy score on testing {:.5f}".format(logistic001.score(X_test,y_test)))
```

```
Accuracy score on training 0.79776
Accuracy score on testing 0.79929
```

In [36]:
```
svm = LinearSVC(max_iter=100000)
svm.fit(X,y)

print("Accuracy score on training {:.5f}".format(svm.score(X_train,y_train)))
print("Accuracy score on testing {:.5f}".format(svm.score(X_test,y_test)))
```

```
Accuracy score on training 0.79776
Accuracy score on testing 0.79929
```

In [37]:
```
svm100 = LinearSVC(C=100, max_iter=100000)
svm100.fit(X,y)

print("Accuracy score on training {:.5f}".format(svm100.score(X_train,y_train)))
print("Accuracy score on testing {:.5f}".format(svm100.score(X_test,y_test)))
```

```
Accuracy score on training 0.79776
Accuracy score on testing 0.79929
```

In [38]:
```
svm001 = LinearSVC(C=0.001, max_iter=100000)
svm001.fit(X,y)

print("Accuracy score on training {:.5f}".format(svm001.score(X_train,y_train)))
print("Accuracy score on testing {:.5f}".format(svm001.score(X_test,y_test)))
```

```
Accuracy score on training 0.79776
Accuracy score on testing 0.79929
```

In [39]:
```
y_pred_logistic = logistic.predict(X_test)
y_pred_logistic
```

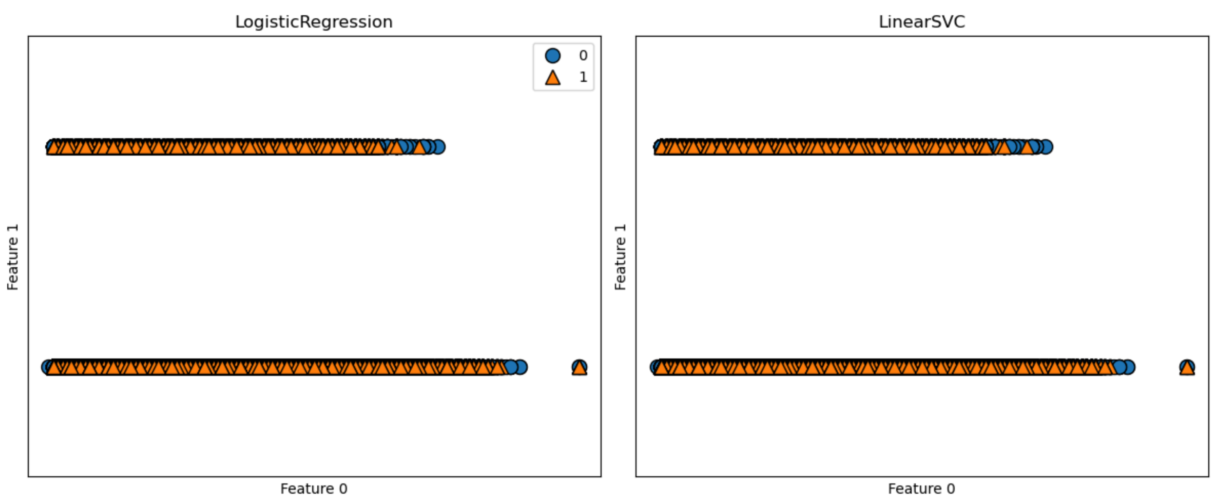
Out[39]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [40]:
```
y_pred_svm = svm.predict(X_test)
y_pred_svm
```

Out[40]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [78]:
```
import mglearn
from mglearn.plot_helpers import discrete_scatter
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
import numpy as np

# Assuming X and y are defined as NumPy arrays
# Select two features for visualization (e.g., columns 0 and 1)
X_plot = X[:, [0, 1]]  # Use NumPy slicing to select two features

# Create a figure with two subplots
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
```

```python
# Define the models
models = [LogisticRegression(max_iter=100000), LinearSVC(max_iter=100000)]

# Iterate over models and axes
for model, ax in zip(models, axes):
    # Fit the model using only the two selected features
    clf = model.fit(X_plot, y)

    # Plot the decision boundary using mglearn
    mglearn.plots.plot_2d_separator(clf, X_plot, fill=False, eps=0.5, ax=ax, alpha=

    # Scatter plot of the data points
    discrete_scatter(X_plot[:, 0], X_plot[:, 1], y, ax=ax)

    # Set title and labels
    ax.set_title("{}".format(clf.__class__.__name__))
    ax.set_xlabel("Feature 0")
    ax.set_ylabel("Feature 1")

# Add a legend to the first subplot
axes[0].legend(loc='best')

# Adjust layout for better spacing
plt.tight_layout()
plt.show()
```



In [ ]: