Project: Analysis of No-Show Medical Appointments

Table of Contents

Introduction

The dataset chosen for this project is [No-Show Appointments (https://www.kaggle.com/joniarroba/noshowappointments)](https://www.kaggle.com/joniarroba/noshowappointments).

The data includes information from 100k medical appointments in Brazil and is focused on the question of whether or not patients show up for their appointment.

We will look at the various characteristics of the patients from the data and try to find correlations between them and the no-show behaviour.

Data Dictionary

PatientId: Unique ID of a patient.

AppointmentID: Unique ID of an appointment.

Gender: Male (M) or Female (F).

ScheduledDay: The day an appointment was scheduled.

AppointmentDay: The day of on which a patient needs to show-up for their scheduled appointment.

Age: Age of the patient.

Neighbourhood: Region of Brazil where an appointment would take place.

Scholarship: True (1) of False (0). A government-sponsored cash-transfer program for poor families.

Hipertension: True (1) or False (0).

Diabetes: True (1) or False (0).

Alcoholism: True (1) or False (0).

Handcap: Number of disabilities in a patient.

SMS_received: True (1) or False (0). Whether the patient received an SMS regarding their appointment

No-show: Yes or No. Yes means the patient did not show up to their appointment. No means the patient showed up to their appointment.

We will try to answer the following questions through this analysis:

1. Does the no-show behaviour occur more in a particular gender?
2. Does the no-show behaviour occur more in a particular age-group?
3. Does having scholarship affect no-show behaviour?
4. Does the no-show behaviour occur more on a particular day?
5. Does waiting period affect no-show behaviour?
6. Does recieving an SMS affect no-show behaviour?
7. Does number of handicaps in a patient affect no-show behaviour?
8. How do different medical conditions correlate with making appointments and not showing up?

In [1]:

```
# Import all libraries that will be used in the project

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from datetime import datetime
```

Data Wrangling

In this section of the report, we will load in the data, check for cleanliness, and then trim and clean the dataset for analysis.

General Properties

In [2]:

```
# Load the dataset and view the first 5 rows

df = pd.read_csv('noshow.csv')
df.head()
```

Out[2]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhoc
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM D PENH
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM D PENH
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA D PRA
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL D CAMBU
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM D PENH

In [3]:

```
# Check number of rows and columns

df.shape
```

Out[3]:

```
(110527, 14)
```

In [4]:

```
# Check data types of all the fields

df.dtypes
```

Out[4]:

```
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 [5]:

```
# Check if there are any null values in a column

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64  
 2   Gender          110527 non-null  object 
 3   ScheduledDay    110527 non-null  object 
 4   AppointmentDay  110527 non-null  object 
 5   Age             110527 non-null  int64  
 6   Neighbourhood   110527 non-null  object 
 7   Scholarship     110527 non-null  int64  
 8   Hipertension    110527 non-null  int64  
 9   Diabetes        110527 non-null  int64  
 10  Alcoholism      110527 non-null  int64  
 11  Handcap         110527 non-null  int64  
 12  SMS_received    110527 non-null  int64  
 13  No-show         110527 non-null  object 
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

Data type of ScheduledDay and AppointmentDay should be datetime. We will change this later

In [6]:

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

Out[6]:

```
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
```

There are no null values in this dataset

In [7]:

```
# Get summary statistics for non-string data types

df.describe()
```

Out[7]:

	PatientId	AppointmentID	Age	Scholarship	Hipertension	Diab
count	1.105270e+05	1.105270e+05	110527.000000	110527.000000	110527.000000	110527.00C
mean	1.474963e+14	5.675305e+06	37.088874	0.098266	0.197246	0.071
std	2.560949e+14	7.129575e+04	23.110205	0.297675	0.397921	0.258
min	3.921784e+04	5.030230e+06	-1.000000	0.000000	0.000000	0.00C
25%	4.172614e+12	5.640286e+06	18.000000	0.000000	0.000000	0.00C
50%	3.173184e+13	5.680573e+06	37.000000	0.000000	0.000000	0.00C
75%	9.439172e+13	5.725524e+06	55.000000	0.000000	0.000000	0.00C
max	9.999816e+14	5.790484e+06	115.000000	1.000000	1.000000	1.00C

Minimum value in Age column is '-1' which is not possible. We will remove this row later.

In [8]:

```
# Check number of unique values in a column

df.nunique()
```

Out[8]:

```
PatientId          62299
AppointmentID     110527
Gender                 2
ScheduledDay      103549
AppointmentDay        27
Age                  104
Neighbourhood         81
Scholarship            2
Hipertension           2
Diabetes               2
Alcoholism             2
Handcap                5
SMS_received           2
No-show                2
dtype: int64
```

In [9]:

```
# Check number of duplicate rows in the dataset

df.duplicated().sum()
```

Out[9]:

```
0
```

There are no duplicate rows in this dataset

Data Cleaning

In [10]:

```
# Get list of all column names in the dataset

df.columns
```

Out[10]:

```
Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hipertens
ion',
       'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'No-show'],
      dtype='object')
```

In [11]:

```
# Removing columns that will not contribute to our analysis

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

df.head(1)
```

Out[11]:

	Gender	ScheduledDay	AppointmentDay	Age	Scholarship	Hipertension	Diabetes	Alcohc
0	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	0	1	0	

In [12]:

```
# Renaming columns to fix typos and making writing subsequent code easier

df.rename(columns = {'ScheduledDay' : 'Scheduled_Day', 'AppointmentDay' : 'Appointment_
Day', 'Hipertension' : 'Hypertension',
                     'Handcap' : 'Handicap', 'SMS_received' : 'SMS_Received', 'No-show'
: 'No_Show'}, inplace = True)

df.head(1)
```

Out[12]:

	Gender	Scheduled_Day	Appointment_Day	Age	Scholarship	Hypertension	Diabetes	Alc
0	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	0	1	0	

In [13]:

```
# Changing all column names to lower case

df.columns = df.columns.str.lower()

df.head(1)
```

Out[13]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcohc
0	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	0	1	0	

In [14]:

```
# Checking index of row with negative value of age

df.query('age == -1')
```

Out[14]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	al
99832	F	2016-06-06T08:58:13Z	2016-06-06T00:00:00Z	-1	0	0	0	

In [15]:

```
# Dropping row with negative age value

df.drop(df.index[99832], inplace = True)

df.shape
```

Out[15]:

```
(110526, 11)
```

In [16]:

```
# Checking if the row removal operation worked

df.query('age == -1')
```

Out[16]:

gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcohol

In [17]:

```
# Confirmation by checking if number of rows has reduced

df.shape
```

Out[17]:

```
(110526, 11)
```

In [18]:

```
# Since there are 104 unique values of Age, grouping them might lead to better visualiz
ations and insights

bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200]

group_names = ['<10', '10s', '20s', '30s', '40s', '50s', '60s', '70s', '80s', '90s', '1
00s']
```

In [19]:

```
# Create new column age_group in existing dataframe

df['age_group'] = pd.cut(df.age, bins, labels = group_names)
```

In [20]:

```
df.head()
```

Out[20]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	0	1	0	
1	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	0	0	0	
2	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	0	0	0	
3	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	0	0	0	
4	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	0	1	1	

In [21]:

```
# Change datatype of scheduled_day and appointment_day to datetime so that we can use d
atetime functions of pandas

df['scheduled_day'] = pd.to_datetime(df.scheduled_day)
df['appointment_day'] = pd.to_datetime(df.appointment_day)
```

In [22]:

```
# Confirm change of datatype

df.dtypes
```

Out[22]:

```
gender                          object
scheduled_day      datetime64[ns, UTC]
appointment_day    datetime64[ns, UTC]
age                              int64
scholarship                      int64
hypertension                     int64
diabetes                         int64
alcoholism                       int64
handicap                         int64
sms_received                     int64
no_show                         object
age_group                     category
dtype: object
```

In [23]:

```
df.head()
```

Out[23]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	0	1	0	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	0	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	0	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	0	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	0	1	1	

In [24]:

```
# Create new column in dataframe with name of day on which an appointment was scheduled

df['scheduled_dayname'] = df.scheduled_day.dt.day_name()
```

In [25]:

```
df.head()
```

Out[25]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	0	1	0	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	0	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	0	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	0	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	0	1	1	

In [26]:

```
# Create new column in dataframe with name of day for which an appointment was schedule
d

df['appointment_dayname'] = df.appointment_day.dt.day_name()
```

In [27]:

```
df.head()
```

Out[27]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	0	1	0	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	0	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	0	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	0	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	0	1	1	

In [28]:

```
# Create new column in dataframe which tells number of days between an appointment and when it was scheduled

df['waiting_period'] = df['appointment_day'] - df['scheduled_day']
```

In [29]:

```
df.head()
```

Out[29]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	0	1	0	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	0	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	0	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	0	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	0	1	1	

We need to extract the day value from waiting_period and change the '-1' value to '0' because it is signifying that the appointment was on the same day as it was scheduled.

In [30]:

```
# Extracting the days value from waiting_period

df['waiting_period'] = round(df['waiting_period'] / np.timedelta64(1, 'D'))
```

In [31]:

```
df.head()
```

Out[31]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	0	1	0	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	0	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	0	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	0	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	0	1	1	

In [32]:

```
# Change datatype to int as we need only absolute value for our analysis

df['waiting_period'] = df['waiting_period'].astype('int')
```

In [33]:

```
df.head()
```

Out[33]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	0	1	0	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	0	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	0	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	0	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	0	1	1	

In [34]:

```
# Putting lower limit to values in the column 'waiting_period' so that any negative val
ue would become 0

df['waiting_period'] = df['waiting_period'].clip(lower = 0)
```

In [35]:

```
df.head()
```

Out[35]:

	gender	scheduled_day	appointment_day	age	scholarship	hypertension	diabetes	alcoho
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	0	1	0	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	0	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	0	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	0	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	0	1	1	

In [36]:

```
# bin edges that will be used to "cut" the data into groups

bin_edges = [-1, 0, 5, 15, 179]

# labels for the four waiting period groups

bin_labels = ['Same Day','1-4 Days','5-14 Days','15 Days and above']

df['days_diff'] = pd.cut(df['waiting_period'], bin_edges, labels=bin_labels)
```

In [37]:

```
# Dropping scheduled_day and appointment_day as we have extracted required information
into new columns

df.drop(columns = ['scheduled_day', 'appointment_day', 'waiting_period'], inplace = Tru
e)
```

In [38]:

```
# Check if desired columns have been dropped

df.head()
```

Out[38]:

	gender	age	scholarship	hypertension	diabetes	alcoholism	handicap	sms_received	no
0	F	62	0	1	0	0	0	0	
1	M	56	0	0	0	0	0	0	
2	F	62	0	0	0	0	0	0	
3	F	8	0	0	0	0	0	0	
4	F	56	0	1	1	0	0	0	

In [39]:

```
# Shortening some column names

df.rename(columns = {'scheduled_dayname' : 'schedule_day', 'appointment_dayname' : 'app
ointment_day'}, inplace = True)
```

In [40]:

```
# Make duplicate of 'handicap' column with the name 'handicap_num'

df['handicap_num'] = df['handicap']

# Change values in 'handicap' column to only convey whether patient had any handicap. W
e now have two columns :
# 'handicap' which tells whether patient has handicap
# 'handicap_num' which tells the number of handicaps in a patient

df['handicap'] = np.where(df['handicap'] > 1, 1, df['handicap'])

df.head()
```

Out[40]:

	gender	age	scholarship	hypertension	diabetes	alcoholism	handicap	sms_received	no
0	F	62	0	1	0	0	0	0	
1	M	56	0	0	0	0	0	0	
2	F	62	0	0	0	0	0	0	
3	F	8	0	0	0	0	0	0	
4	F	56	0	1	1	0	0	0	

In [41]:

```
# Make column 'multiple_condition' which would be used to convey if a patient has multi
ple conditions

df['multiple_condition'] = df['alcoholism'] + df['hypertension'] + df['diabetes'] + df
['handicap']

# 'num_condition' will tell the total number of conditions a patient has

df['num_condition'] = df['multiple_condition']

df['multiple_condition'] = np.where(df['multiple_condition'] > 1 , 1, 0)

df.head()
```

Out[41]:

	gender	age	scholarship	hypertension	diabetes	alcoholism	handicap	sms_received	no
0	F	62	0	1	0	0	0	0	
1	M	56	0	0	0	0	0	0	
2	F	62	0	0	0	0	0	0	
3	F	8	0	0	0	0	0	0	
4	F	56	0	1	1	0	0	0	

In [42]:

```
# Reordering the columns for better readability

df = df[['gender', 'age', 'age_group', 'schedule_day', 'appointment_day', 'days_diff', 
'sms_received', 'no_show',
         'scholarship', 'alcoholism', 'hypertension', 'diabetes', 'handicap', 'handicap
_num', 'multiple_condition', 'num_condition']]

df.head()
```

Out[42]:

	gender	age	age_group	schedule_day	appointment_day	days_diff	sms_received	no_sh
0	F	62	60s	Friday	Friday	Same Day	0	N
1	M	56	50s	Friday	Friday	Same Day	0	N
2	F	62	60s	Friday	Friday	Same Day	0	N
3	F	8	<10	Friday	Friday	Same Day	0	N
4	F	56	50s	Friday	Friday	Same Day	0	N

In [43]:

```
# Replacing the values in no_show column as existing format is confusing to interpret

df['no_show'] = df['no_show'].replace({'No' : 0, 'Yes' : 1})
```

In [44]:

```
df.head()
```

Out[44]:

	gender	age	age_group	schedule_day	appointment_day	days_diff	sms_received	no_sh
0	F	62	60s	Friday	Friday	Same Day	0	
1	M	56	50s	Friday	Friday	Same Day	0	
2	F	62	60s	Friday	Friday	Same Day	0	
3	F	8	<10	Friday	Friday	Same Day	0	
4	F	56	50s	Friday	Friday	Same Day	0	

Exploratory Data Analysis

In [45]:

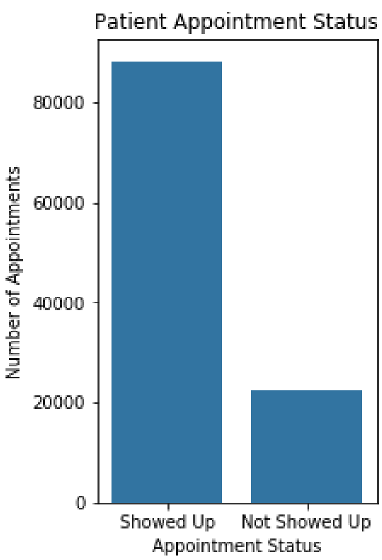
```
# Define function to make Count Plot as code would be repetitive

def count_plot(df_data, x_data, x_label, y_label, plot_title, plot_color = 'tab:blue', plot_palette = None):
    plot = sns.countplot(data = df_data, x = x_data, color = plot_color, palette = plot_palette)
    plt.title(plot_title)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    return plot
```

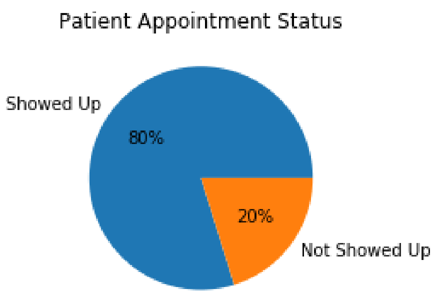
In [46]:

```
# Custom Figure Size to accomodate the two subplots

plt.figure(figsize = [10,5])

plt.subplot(1,3,1)

# Plotting Bar Chart

count_plot(df, 'no_show', 'Appointment Status', 'Number of Appointments', 'Patient Appointment Status' )
plt.xticks([0,1,],['Showed Up', 'Not Showed Up'])

plt.subplot(1,3,3)

# Plotting a Pie Chart

plt.pie(df.no_show.value_counts(), labels = ['Showed Up', 'Not Showed Up'], autopct='%1.0f%%')
plt.title('Patient Appointment Status')
```

Out[46]:

```
Text(0.5, 1.0, 'Patient Appointment Status')
```





In 80% of appointments, the patient showed up as scheduled.

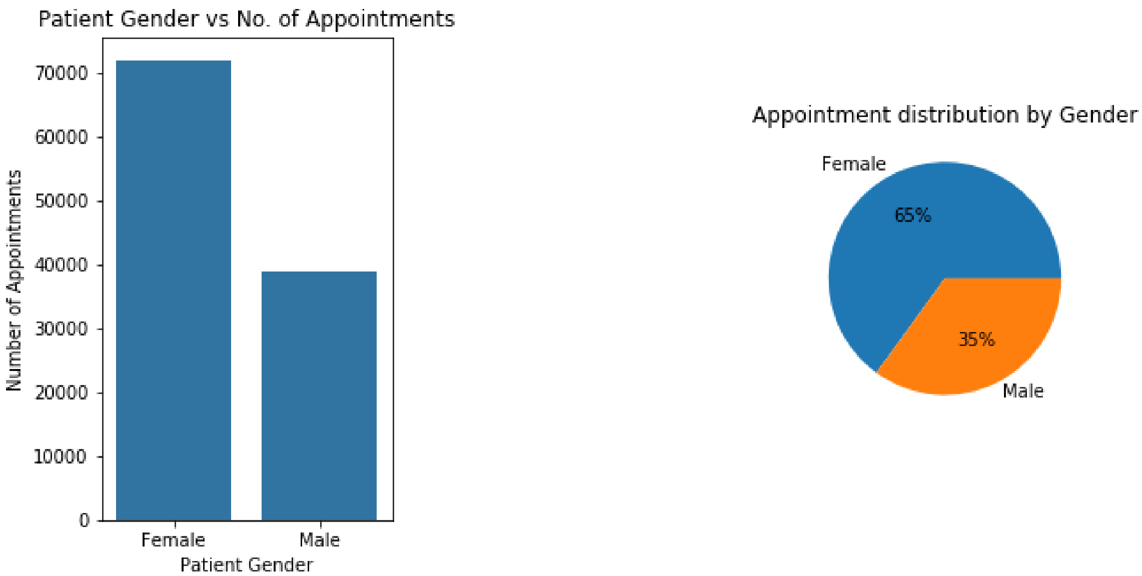
Q1. Does the no-show behaviour occur more in a particular gender?

In [47]:

```
plt.figure(figsize = [10,5])

plt.subplot(1,3,1)

count_plot(df, 'gender', 'Patient Gender', 'Number of Appointments', 'Patient Gender vs No. of Appointments' )
plt.xticks([0,1,],['Female', 'Male'])

plt.subplot(1,3,3)

plt.pie(df.gender.value_counts(), labels = ['Female', 'Male'], autopct='%1.0f%%')
plt.title('Appointment distribution by Gender')
```

Out[47]:

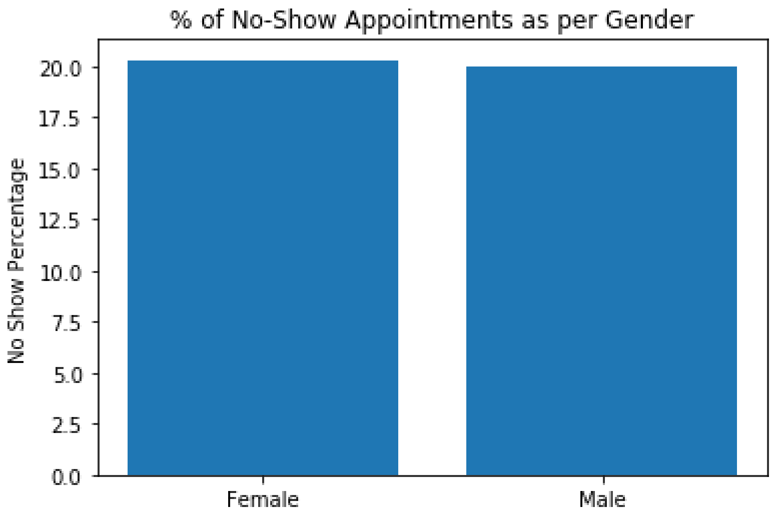
```
Text(0.5, 1.0, 'Appointment distribution by Gender')
```



65% of the appointments were made by females.

In [48]:

```
# Obtaining a series with % of counts who did not show up for appointment as per gender

ns_g = df.groupby('gender').no_show.mean()*100

# Plotting the series obtained in a bar chart

plt.bar(x = ns_g.index, height = ns_g)
plt.title('% of No-Show Appointments as per Gender')
plt.xticks([0, 1], ['Female', 'Male'])
plt.ylabel('No Show Percentage');
```



~20% of Females and Males don't show up to their scheduled appointments.

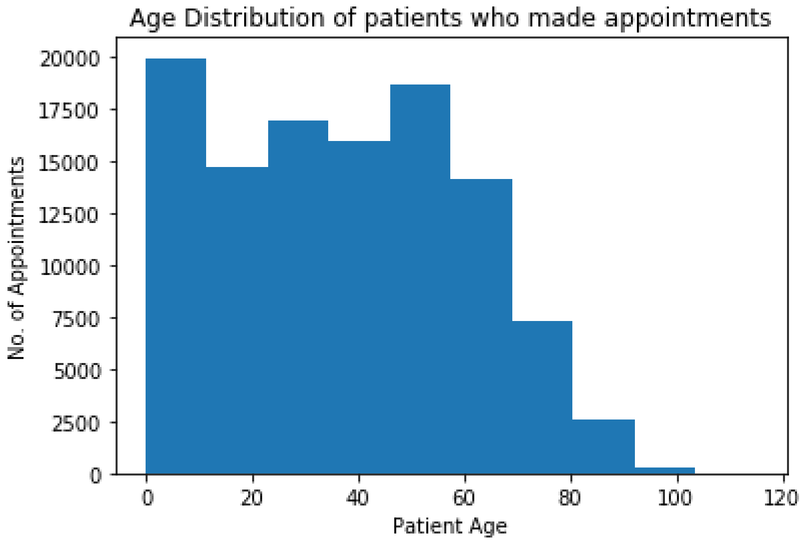
Q2. Does the no-show behaviour occur more in a particular age-group?

In [49]:

```
# Plotting a Histogram to observe distribution of age of patients w.r.t no. of appointm
ents

plt.hist(data = df, x = 'age')
plt.xlabel('Patient Age')
plt.ylabel('No. of Appointments')
plt.title('Age Distribution of patients who made appointments')
```

Out[49]:

```
Text(0.5, 1.0, 'Age Distribution of patients who made appointments')
```



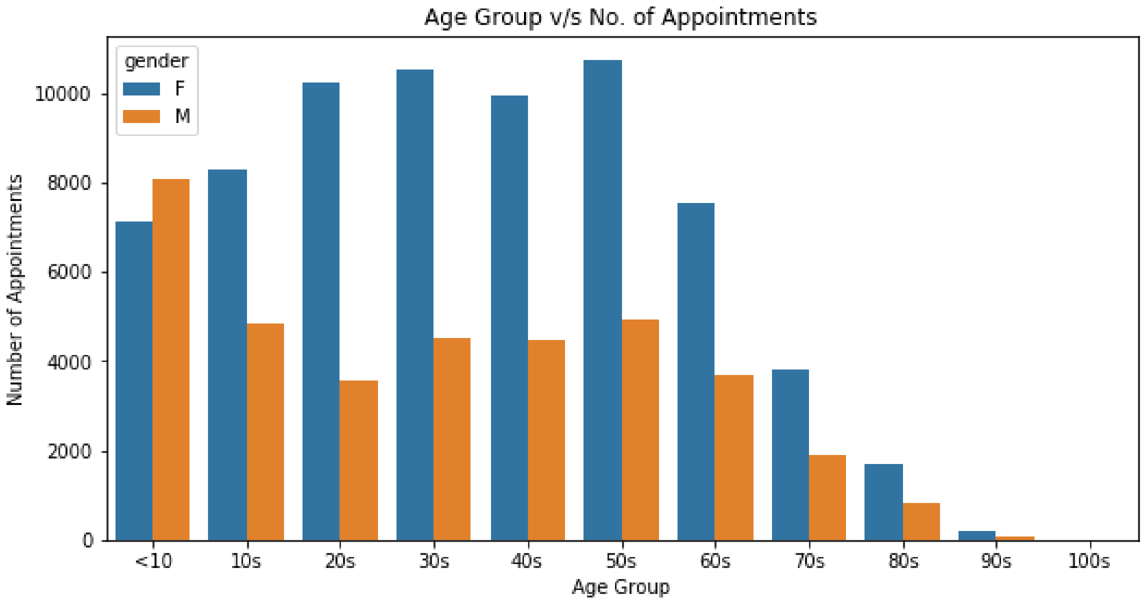
Younger patients tend to make more appointments.

In [50]:

```
# Plotting a Bar Chart which would help us compare no. of appointments made by patients of different age groups
# in the two genders

plt.figure(figsize = [10,5])

sns.countplot(data = df, x = 'age_group',  hue = 'gender')
plt.title('Age Group v/s No. of Appointments')
plt.xlabel('Age Group')
plt.ylabel('Number of Appointments')
```

Out[50]:

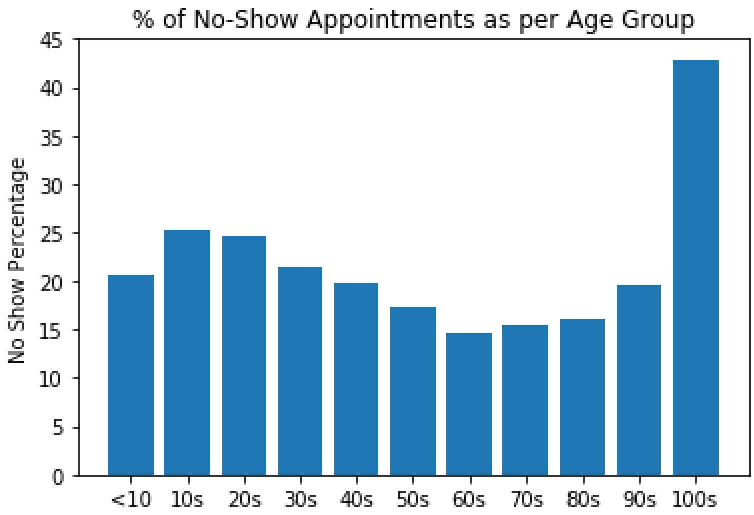
```
Text(0, 0.5, 'Number of Appointments')
```



Interestingly, highest number of appointments among females were made in the age groups from 20s through 50s, whereas highest number of appointments among males were made in the age group of less than 10.

In [51]:

```
# Plotting Bar Chart to show % of no-shows for different Age Groups

ns_age = df.groupby('age_group').no_show.mean()*100

plt.bar(x = ns_age.index, height = ns_age)
plt.title('% of No-Show Appointments as per Age Group')
plt.ylabel('No Show Percentage');
```



No-Show Appointments were more prominent among patients above 100 years of age, but for other age groups it is lying between 15-25%. The least percentage of no-show appointments is seen among patients of less than 10 years in age (probably because they would be under the care of an adult), and patients from 50s to 80s, suggesting that there is a correlation between a patient's age and their probability of showing up to an appointment.

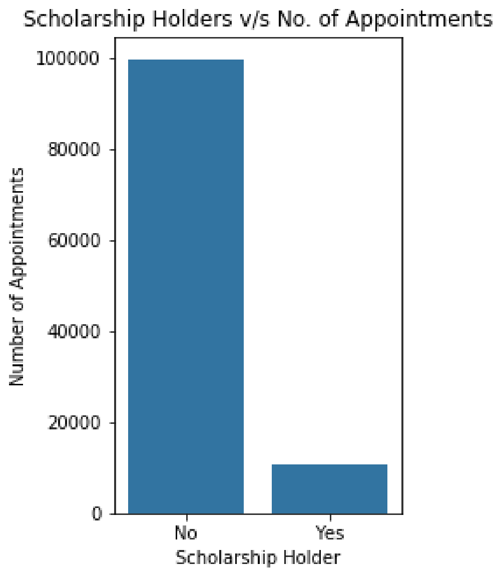
Q3. Does having scholarship affect no-show behaviour?

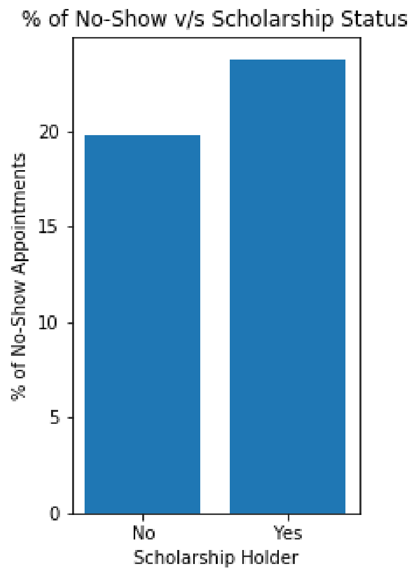
In [52]:

```
plt.figure(figsize = [10,5])

plt.subplot(1,3,1)

count_plot(df, 'scholarship', 'Scholarship Holder', 'Number of Appointments', 'Scholars
hip Holders v/s No. of Appointments' )
plt.xticks([0,1], ['No', 'Yes'])


plt.subplot(1,3,3)

ns_s = df.groupby('scholarship').no_show.mean()*100
plt.bar(x = ns_s.index, height = ns_s)
plt.title('% of No-Show v/s Scholarship Status')
plt.xlabel('Scholarship Holder')
plt.xticks([0,1], ['No', 'Yes'])
plt.ylabel('% of No-Show Appointments')
```

Out[52]:

```
Text(0, 0.5, '% of No-Show Appointments')
```





Majority of the appointments were made by patients who were not on scholarship. Also, the % of no-show is higher among patients who had scholarship. Hence, there is a correlation between scholarship status and no-show behaviour.

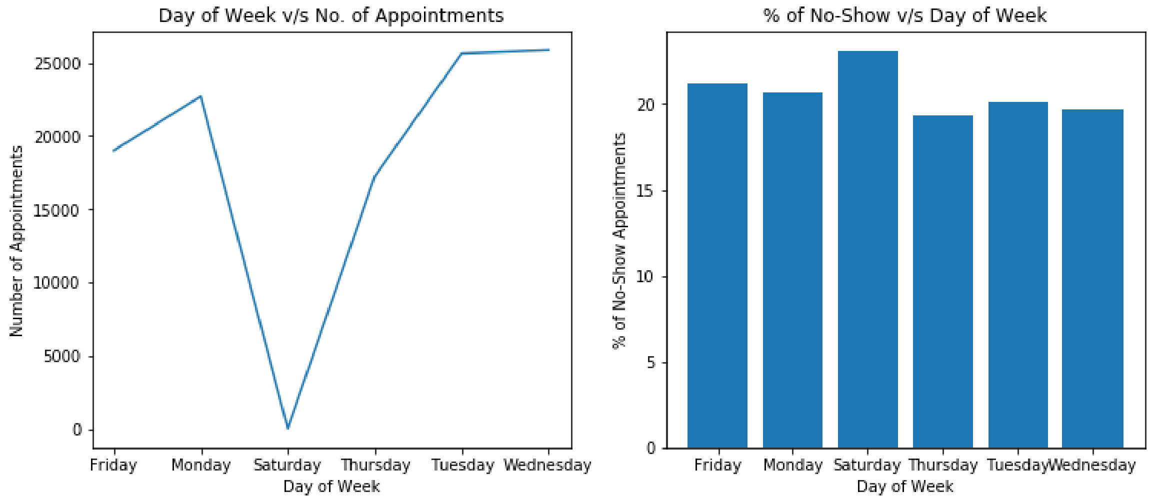
Q4. Does the no-show behaviour occur more on a particular day?

In [53]:

```
plt.figure(figsize = [19,5])

days_count =  df.appointment_day.value_counts()

plt.subplot(1,3,1)

# Plotting a Line Plot to observe how no. of appointments rise and dip throughout the w
eek

sns.lineplot(x = days_count.index, y = days_count)
plt.title('Day of Week v/s No. of Appointments')
plt.xlabel('Day of Week')
plt.ylabel('Number of Appointments')

plt.subplot(1,3,2)

ns_d = df.groupby('appointment_day').no_show.mean()*100

plt.bar(x = ns_d.index, height = ns_d)
plt.title('% of No-Show v/s Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('% of No-Show Appointments')
```

Out[53]:

```
Text(0, 0.5, '% of No-Show Appointments')
```



In [54]:

```
# Listing the values here so that any confusion due to the days not being arranged as t
hey are in the week on the graph x-axis

df.appointment_day.value_counts()
```

Out[54]:

```
Wednesday    25867
Tuesday      25640
Monday       22714
Friday       19019
Thursday     17247
Saturday        39
Name: appointment_day, dtype: int64
```

In [55]:

```
df.groupby('appointment_day').no_show.mean()*100
```

Out[55]:

```
appointment_day
Friday       21.226142
Monday       20.648058
Saturday     23.076923
Thursday     19.354091
Tuesday      20.093604
Wednesday    19.689179
Name: no_show, dtype: float64
```

There were no appointments made on Sunday. It can be seen that patients prefer to make appointments for weekdays, with the most appointments scheduled on Wednesday and the least on Saturday.

Also, highest no-show percentage were for appointments on Saturday, Friday, and Monday i.e. the days near and including the weekends.

Hence, we can see there is a correlation between day of the week and possibility of patient not showing up to their appoitnment.

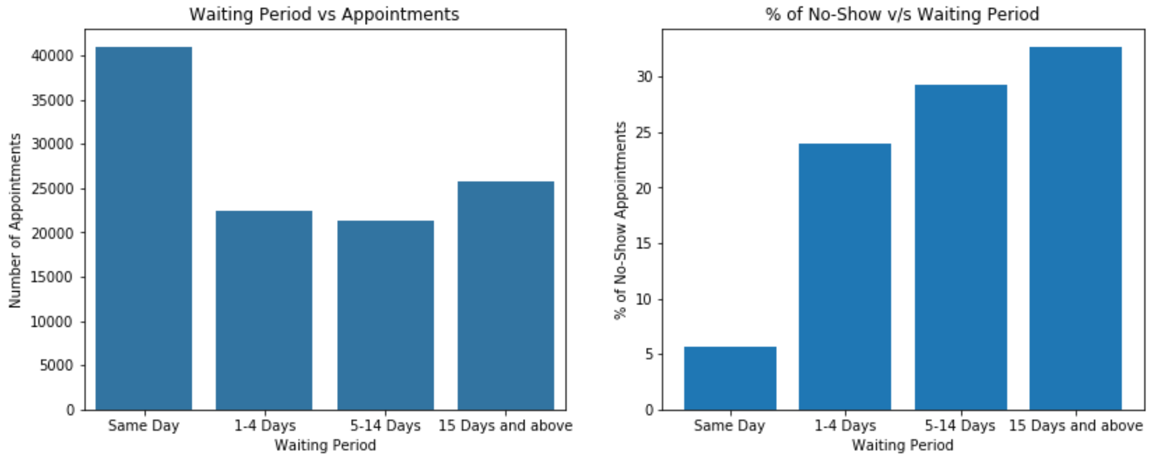
Q5. Does waiting period affect no-show behaviour?

In [56]:

```
plt.figure(figsize = [21,5])

plt.subplot(1,3,1)

count_plot(df, 'days_diff', 'Waiting Period', 'Number of Appointments', 'Waiting Period vs Appointments' )


plt.subplot(1,3,2)

ns_w = df.groupby('days_diff').no_show.mean()*100
plt.bar(x = ns_w.index, height = ns_w)
plt.title('% of No-Show v/s Waiting Period')
plt.xlabel('Waiting Period')
plt.ylabel('% of No-Show Appointments')
```

Out[56]:

```
Text(0, 0.5, '% of No-Show Appointments')
```



It is evident that patients like to schedule their appointments on the same day, and are more unlikely to show up as the waiting period increases.

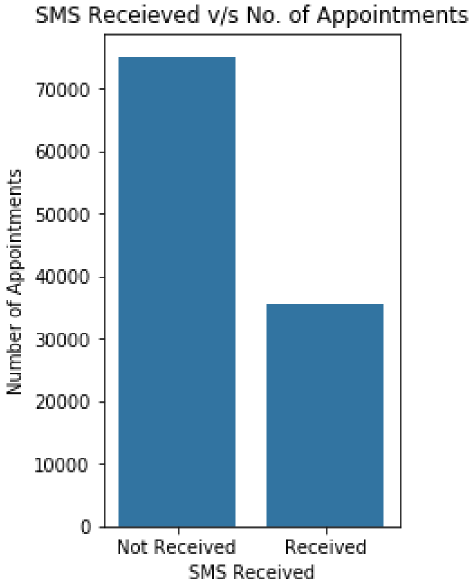
Q6. Does recieving an SMS affect no-show behaviour?

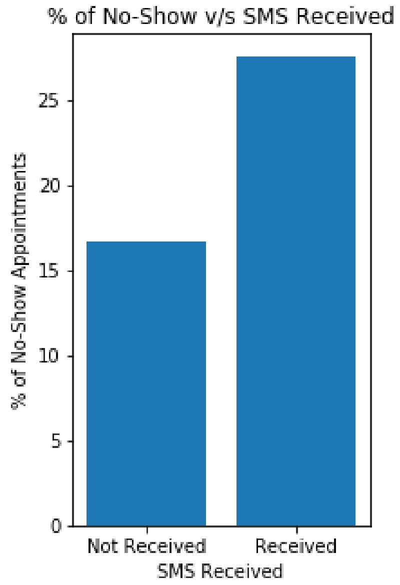
In [57]:

```
plt.figure(figsize = [10,5])

plt.subplot(1,3,1)

count_plot(df, 'sms_received', 'SMS Received', 'Number of Appointments', 'SMS Receieved v/s No. of Appointments' )
plt.xticks([0,1], ['Not Received', 'Received'])

plt.subplot(1,3,3)

ns_m = df.groupby('sms_received').no_show.mean()*100
plt.bar(x = ns_m.index, height = ns_m)
plt.title('% of No-Show v/s SMS Received')
plt.xlabel('SMS Received')
plt.xticks([0,1], ['Not Received', 'Received'])
plt.ylabel('% of No-Show Appointments')
```

Out[57]:

```
Text(0, 0.5, '% of No-Show Appointments')
```





It is interesting to note that while most of the appointments did not receive an SMS, the no. of appointments who did receive an SMS and did not show up was higher.

Hence, we can say that if a patient receives an SMS, they are more likely to miss their appointment.

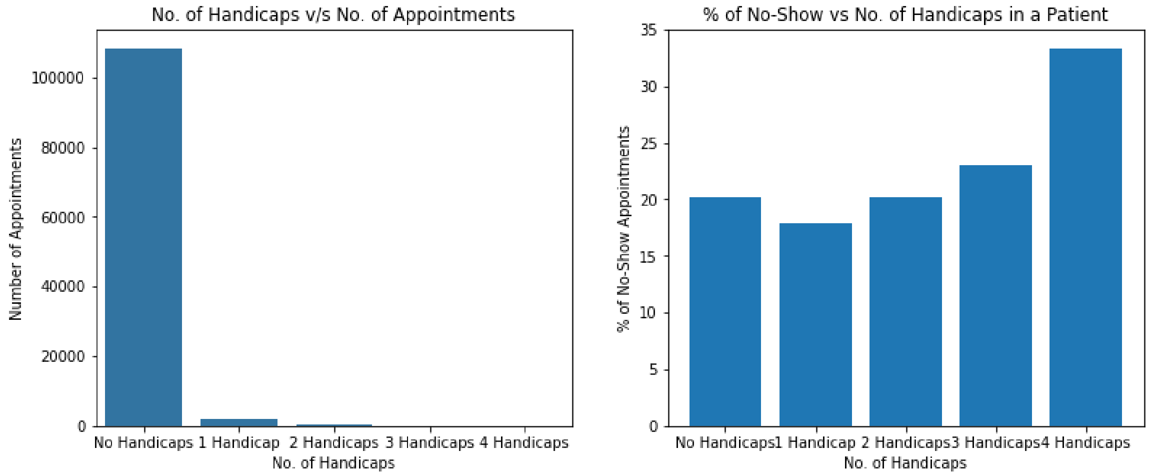
Q7. Does number of handicaps in a patient affect no-show behaviour?

In [58]:

```
plt.figure(figsize = [20,5])

plt.subplot(1,3,1)

count_plot(df, 'handicap_num', 'No. of Handicaps', 'Number of Appointments', 'No. of Ha
ndicaps v/s No. of Appointments' )
plt.xticks([0,1,2,3,4], ['No Handicaps', '1 Handicap', '2 Handicaps', '3 Handicaps', '4
Handicaps'])

plt.subplot(1,3,2)

ns_h = df.groupby('handicap_num').no_show.mean()*100
plt.bar(x = ns_h.index, height = ns_h)
plt.title('% of No-Show vs No. of Handicaps in a Patient')
plt.xlabel('No. of Handicaps')
plt.xticks([0,1,2,3,4], ['No Handicaps', '1 Handicap', '2 Handicaps', '3 Handicaps', '4
Handicaps'])
plt.ylabel('% of No-Show Appointments')
```

Out[58]:

```
Text(0, 0.5, '% of No-Show Appointments')
```



In [59]:

```
df.handicap_num.value_counts()
```

Out[59]:

```
0    108285
1      2042
2       183
3        13
4         3
Name: handicap_num, dtype: int64
```

There is huge gap between the number of appointments made by handicapped and non-handicapped people, with the number being relatively miniscule for patients with multiple handicaps.

But, we can see that the % of no-show appointments increases with the number of handicaps.

Hence, we can say that the more the number of handicaps in a patient, the more their chances of missing their appointment.

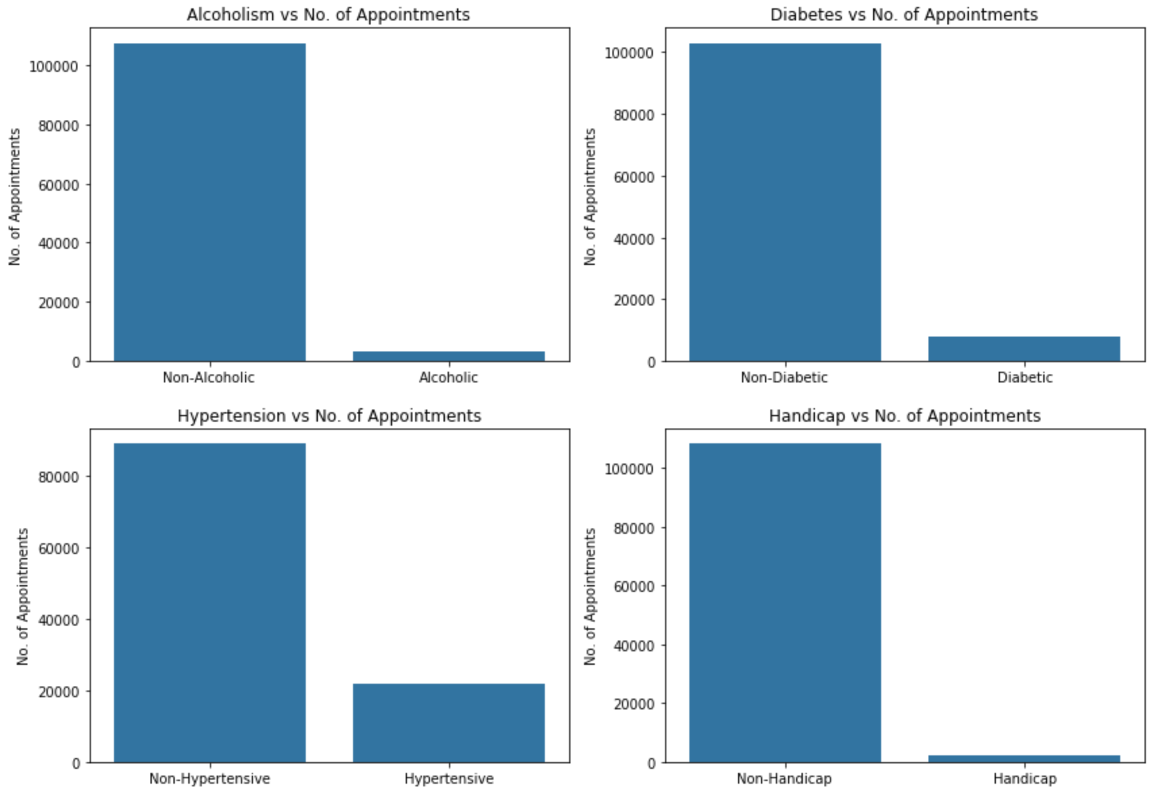
Q8. How do different medical conditions correlate with making appointments and not showing up?

In [60]:

```
plt.figure(figsize = [14, 10])

# alcoholism

plt.subplot(2, 2, 1)

count_plot(df, 'alcoholism', '', 'No. of Appointments', 'Alcoholism vs No. of Appointme
nts' )
plt.xticks([0, 1], ['Non-Alcoholic', 'Alcoholic'])


# diabetes

plt.subplot(2, 2, 2)

count_plot(df, 'diabetes', '', 'No. of Appointments', 'Diabetes vs No. of Appointment
s')
plt.xticks([0, 1], ['Non-Diabetic', 'Diabetic'])

# hypertension

plt.subplot(2, 2, 3)

count_plot(df, 'hypertension', '', 'No. of Appointments', 'Hypertension vs No. of Appoi
ntments' )
plt.xticks([0, 1], ['Non-Hypertensive', 'Hypertensive'])

# handicap

plt.subplot(2, 2, 4)

count_plot(df, 'handicap', '', 'No. of Appointments', 'Handicap vs No. of Appointments'
)
plt.xticks([0, 1], ['Non-Handicap', 'Handicap'])
```

Out[60]:

```
([<matplotlib.axis.XTick at 0x1cce4363fc8>,
  <matplotlib.axis.XTick at 0x1cce4363348>],
 <a list of 2 Text xticklabel objects>)
```



In [61]:

```
# Declare empty dictionary where the percentages of no-show appointments for each medic
al condition would be stored

cond_dict = {}

# percentage of no-show appointments by patients who were only alcoholics

cond_dict['alcoholism'] = (df[df.num_condition <= 1].groupby('alcoholism').no_show.mean
() * 100)[1]

# percentage of no-show appointments by patients who were only hypertensive

cond_dict['hypertension'] = (df[df.num_condition <= 1].groupby('hypertension').no_show.
mean() * 100)[1]

# percentage of no-show appointments by patients who were only diabetic

cond_dict['diabetes'] = (df[df.num_condition <= 1].groupby('diabetes').no_show.mean() *
100)[1]

# percentage of no-show appointments by patients who were only handicap

cond_dict['handicap'] = (df[df.num_condition <= 1].groupby('handicap').no_show.mean() *
100)[1]

# percentage of no-show appointments by patients who had multiple conditions

cond_dict['multiple_condition'] = (df.groupby('multiple_condition').no_show.mean() * 10
0)[1]

# View the populated list

cond_dict
```

Out[61]:

```
{'alcoholism': 21.644120707596255,
 'hypertension': 17.016760594305786,
 'diabetes': 20.3579418344519,
 'handicap': 19.632414369256473,
 'multiple_condition': 17.69815418023887}
```

In [62]:

```
# Converting the dictionary to a pandas series

cond_series = pd.Series(cond_dict)

# Sorting the values in descending order for Bar Chart

cond_series.sort_values(ascending=False, inplace=True)
cond_series
```

Out[62]:

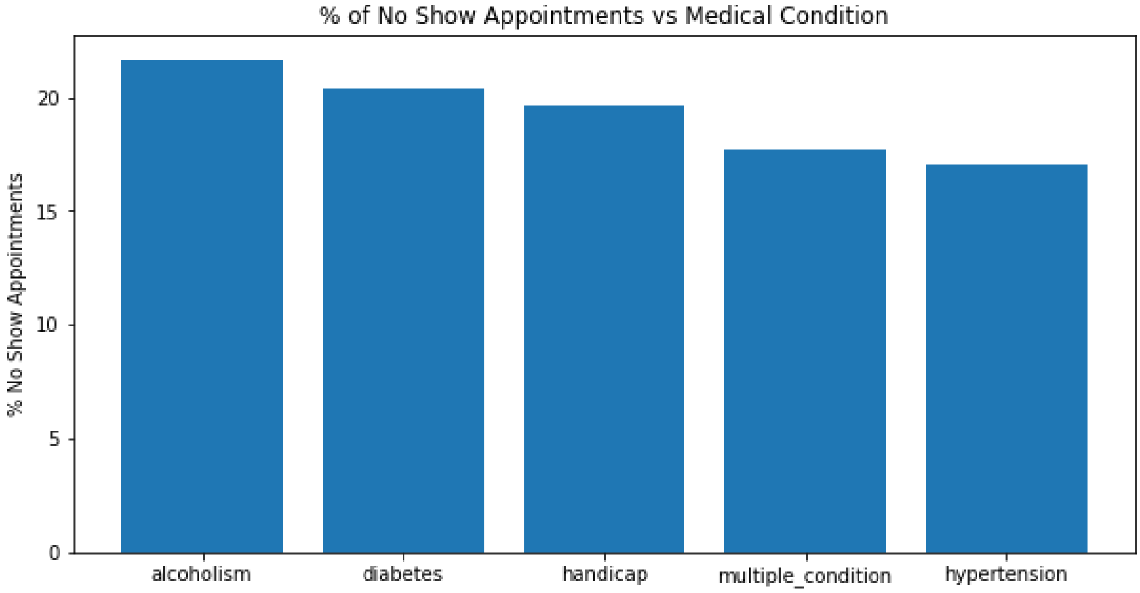
```
alcoholism            21.644121
diabetes              20.357942
handicap              19.632414
multiple_condition    17.698154
hypertension          17.016761
dtype: float64
```

In [63]:

```
plt.figure(figsize = [10,5])

plt.bar(x = cond_series.index, height = cond_series)
plt.title('% of No Show Appointments vs Medical Condition')
plt.ylabel('% No Show Appointments')
```

Out[63]:

```
Text(0, 0.5, '% No Show Appointments')
```



It is evident that the the number of appointments made by patients without any conditions was much higher than those with conditions.

Also, the highest % of no-show cases is among alcoholics, with 21.64% not showing up to their scheduled appointments.

Hence, alcoholics are most likely to not show-up to a scheduled appointment among patients with any medical condition.

Conclusions

Results

- In 80% of appointments, the patient showed up as scheduled
- 65% of the appointments were made by females.
- Younger patients tend to make more appointments.
- Highest number of appointments among females were made in the age groups from 20s through 50s, whereas highest number of appointments among males were made in the age group of less than 10.
- Majority of the appointments were made by patients who were not on scholarship.
- Patients like to schedule their appointments on the same day.
- Most of the appointments did not receive an SMS.

1. Does the no-show behaviour occur more in a particular gender?

~20% of Females and Males don't show up to their scheduled appointments.

Since the value is almost same, we can say there is no correlation between gender and no-show behaviour.

2. Does the no-show behaviour occur more in a particular age-group?

No-Show Appointments were more prominent among patients above 100 years of age, but for other age groups it is lying between 15-25%.

The least percentage of no-show appointments is seen among patients of less than 10 years in age (probably because they would be under the care of an adult), and patients from 50s to 80s, suggesting that there is a correlation between a patient's age and their probability of showing up to an appointment.

3. Does having scholarship affect no-show behaviour?

% of no-show is higher among patients who had scholarship. Hence, there is a correlation between scholarship status and no-show behaviour.

4. Does the no-show behaviour occur more on a particular day?

Highest no-show percentage were for appointments on Saturday, Friday, and Monday i.e. the days near and including the weekends.

Hence, we can see there is a correlation between day of the week and possibility of patient not showing up to their appoitnment.

5. Does waiting period affect no-show behaviour?

Patients are more unlikely to show up as the waiting period increases.

6. Does recieving an SMS affect no-show behaviour?

It is interesting to note that while most of the appointments did not receive an SMS, the no. of appointments who did receive an SMS and did not show up was higher.

Hence, we can say that if a patient receives an SMS, they are more likely to miss their appointment.

7. Does number of handicaps in a patient affect no-show behaviour?

There is huge gap between the number of appointments made by handicapped and non-handicapped people, with the number being relatively miniscule for patients with multiple handicaps.

But, we can see that the % of no-show appointments increases with the number of handicaps.

Hence, we can say that the more the number of handicaps in a patient, the more their chances of missing their appointment.

8. How do different medical conditions correlate with making appointments and not showing up?

It is evident that the the number of appointments made by patients without any conditions was much higher than those with conditions.

Also, the highest % of no-show cases is among alcoholics, with 21.64% not showing up to their scheduled appointments.

Hence, alcoholics are most likely to not show-up to a scheduled appointment among patients with any medical condition.

Limitations

- Since most of our data is categorical and boolean, we cannot investigate further using statistical techniques.
- Some fields could have had more information, eg - sms_received. If we had information about the circumstances around which an SMS is sent (reminder for appointment if the patient did not show up, or reminder before an appointment), we might have made a more informed conclusion since the current one is counter-intuitive.
- The dataset only covers 2016 data. It would be interesting to observe different trends over the years for more solid conclusions.
- Some columns were not described properly and had to be interpreted based on responses on Kaggle discussion forums.

References

https://www.youtube.com/playlist?list=PL5-da3qGB5ICCsgW1MxlZ0Hq8LL5U3u9y (https://www.youtube.com/playlist?list=PL5-da3qGB5ICCsgW1MxlZ0Hq8LL5U3u9y)

https://youtube.com/playlist?list=PLBfyvFO_aKGRaJmdo501Hu_wXwgmjbR50 (https://youtube.com/playlist?list=PLBfyvFO_aKGRaJmdo501Hu_wXwgmjbR50)

https://www.freecodecamp.org/news/how-to-use-timedelta-objects-in-python/ (https://www.freecodecamp.org/news/how-to-use-timedelta-objects-in-python/)

https://towardsdatascience.com/customizing-multiple-subplots-in-matplotlib-a3e1c2e099bc (https://towardsdatascience.com/customizing-multiple-subplots-in-matplotlib-a3e1c2e099bc)

https://www.geeksforgeeks.org/how-to-drop-rows-in-pandas-dataframe-by-index-labels/ (https://www.geeksforgeeks.org/how-to-drop-rows-in-pandas-dataframe-by-index-labels/)

https://stackoverflow.com/ (https://stackoverflow.com/)