Investigate a Dataset (Medical Appointments No Shows)

May 4, 2019

1 Project: Investigate a Dataset (Medical Appointments No Shows)

Table of Contents 1)Introduction

Introduction

In this project I have investigated a dataset of appoinment records for Brasil public hospitals. The data includes some attributes of patients and state if the patients showed up to appointments. The analysis is focused on finding trends influencing patients to show or not show up to appointments.

The original problem description and data set can be found here: https://www.kaggle.com/joniarroba/noshowappointments/home

Dataset Description

Data Wrangling

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        import warnings
        warnings.filterwarnings('ignore')

        %matplotlib inline

In [2]: # Load data to a dataframe
        df = pd.read_csv('noshowappointments-kagglev2-may-2016.csv')

        # Learn the size of the dataset
        df.shape
```

```
Out[2]: (110527, 14)

In [3]: df.columns

Out[3]: Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
               'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hipertension',
               'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'No-show'],
              dtype='object')

In [4]: # Typos in the column names as well as their format should be corrected / unified
        df.columns = ['patient_id', 'appointment_id', 'gender', 'scheduled_day',
                      'appointment_day', 'age', 'neighbourhood', 'scholarship', 'hypertension',
                      'diabetes', 'alcoholism', 'handicap', 'sms_received', 'no_show']
        df.columns

Out[4]: Index(['patient_id', 'appointment_id', 'gender', 'scheduled_day',
               'appointment_day', 'age', 'neighbourhood', 'scholarship',
               'hypertension', 'diabetes', 'alcoholism', 'handicap', 'sms_received',
               'no_show'],
              dtype='object')

In [5]: # Let's have an initial view on the data
        df.head(5)

Out[5]:      patient_id  appointment_id gender         scheduled_day  \
        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

                appointment_day  age      neighbourhood  scholarship  hypertension  \
        0  2016-04-29T00:00:00Z   62    JARDIM DA PENHA            0             1
        1  2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0             0
        2  2016-04-29T00:00:00Z   62      MATA DA PRAIA            0             0
        3  2016-04-29T00:00:00Z    8  PONTAL DE CAMBURI            0             0
        4  2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0             1

           diabetes  alcoholism  handicap  sms_received no_show
        0         0           0         0             0      No
        1         0           0         0             0      No
        2         0           0         0             0      No
        3         0           0         0             0      No
        4         1           0         0             0      No

In [6]: # And another view on the dataset
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
```

```
Data columns (total 14 columns):
patient_id         110527 non-null float64
appointment_id     110527 non-null int64
gender             110527 non-null object
scheduled_day      110527 non-null object
appointment_day    110527 non-null object
age                110527 non-null int64
neighbourhood      110527 non-null object
scholarship        110527 non-null int64
hypertension       110527 non-null int64
diabetes           110527 non-null int64
alcoholism         110527 non-null int64
handicap           110527 non-null int64
sms_received       110527 non-null int64
no_show            110527 non-null object
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

Here are some initial observations:

1)There are 13 independent variables and one dependent (no_show) in the dataset

2)The dataset does not contain any missing values (NaNs).

3)The patient_id data type is float but should be int.

4)The scheduled_day and appointment_day columns type should be changed to datetime.

5)The appointment_day has no hour specified (it equals to 00:00:00). We will not be able to analyze if the appointment hour has anything to do with no shows.

6)There could be interesting to know how much time passed between a visit scheduling time and the actual visit time. There is no such data column but this can be calculated from scheduled_day and appointment_day columns.

7)Another interesting question would be how show and no-show appointments are distributed among days of week. To explore this I will calculate a column called appointment_dow.

Observation 3: The patient_id data type is float but should be int

```
In [7]: # Check how many patients_ids are not integers
        non_int_patient_ids = df[~ df.patient_id.apply(lambda x: x.is_integer())]
        print('There are {} patients_ids that are not integers'.format(len(non_int_patient_ids))
        non_int_patient_ids
```

```
There are 5 patients_ids that are not integers
```

```
Out[7]:           patient_id  appointment_id gender         scheduled_day  \
        3950     93779.52927         5712759      F  2016-05-18T09:12:29Z
        73228   537615.28476         5637728      F  2016-04-29T07:19:57Z
        73303   141724.16655         5637648      M  2016-04-29T07:13:36Z
        100517   39217.84439         5751990      F  2016-05-31T10:56:41Z
        105430   43741.75652         5760144      M  2016-06-01T14:22:58Z

                      appointment_day  age   neighbourhood  scholarship  hypertension  \
```

```
3950    2016-05-18T00:00:00Z  33          CENTRO            0             0
73228   2016-05-06T00:00:00Z  14  FORTE SÃO JOÃO            0             0
73303   2016-05-02T00:00:00Z  12  FORTE SÃO JOÃO            0             0
100517  2016-06-03T00:00:00Z  44    PRAIA DO SUÁ            0             0
105430  2016-06-01T00:00:00Z  39     MARIA ORTIZ            0             0

        diabetes  alcoholism  handicap  sms_received no_show
3950           0           0         0             0      No
73228          0           0         0             1      No
73303          0           0         0             0      No
100517         0           0         0             0      No
105430         1           0         0             0      No
```

As there are only 5 float patient_ids, it seems they are typos. I will check if they would be unique ids when the decimal part is truncated. If yes, I will truncate their decimal part and keep them in the dataset.

```
In [8]: # Extract float patient_ids from the list above
        patient_ids = [93779.52927, 537615.28476, 141724.16655, 39217.84439, 43741.75652]

        # Convert all float patient_ids to int (by truncating the decimal part)
        # and check if such patients exist in the rest of the dataset
        for i in range(len(patient_ids)):
            patient_ids[i] = int(patient_ids[i])
            if df.query('patient_id == {}'.format(patient_ids[i])).empty:
                print('Patient id == {} does not exist.'.format(patient_ids[i]))
            else:
                print('Patient id == {} already exists.'.format(patient_ids[i]))
```

```
Patient id == 93779 does not exist.
Patient id == 537615 does not exist.
Patient id == 141724 does not exist.
Patient id == 39217 does not exist.
Patient id == 43741 does not exist.
```

```
In [9]: # Convert patient_id from float to int
        df['patient_id'] = df['patient_id'].astype('int64')

        # Check if the patient_id is int64
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
patient_id         110527 non-null int64
appointment_id     110527 non-null int64
gender             110527 non-null object
scheduled_day      110527 non-null object
```

```
appointment_day    110527 non-null object
age                110527 non-null int64
neighbourhood      110527 non-null object
scholarship        110527 non-null int64
hypertension       110527 non-null int64
diabetes           110527 non-null int64
alcoholism         110527 non-null int64
handicap           110527 non-null int64
sms_received       110527 non-null int64
no_show            110527 non-null object
dtypes: int64(9), object(5)
memory usage: 11.8+ MB
```

Observation 4: The scheduled_day and appointment_day columns type should be changed to datetime

In [10]:
```
# Convert columns types
df['scheduled_day'] = pd.to_datetime(df['scheduled_day']).dt.date.astype('datetime64[ns
df['appointment_day'] = pd.to_datetime(df['appointment_day']).dt.date.astype('datetime6

# Check if the type is now datetime
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
patient_id         110527 non-null int64
appointment_id     110527 non-null int64
gender             110527 non-null object
scheduled_day      110527 non-null datetime64[ns]
appointment_day    110527 non-null datetime64[ns]
age                110527 non-null int64
neighbourhood      110527 non-null object
scholarship        110527 non-null int64
hypertension       110527 non-null int64
diabetes           110527 non-null int64
alcoholism         110527 non-null int64
handicap           110527 non-null int64
sms_received       110527 non-null int64
no_show            110527 non-null object
dtypes: datetime64[ns](2), int64(9), object(3)
memory usage: 11.8+ MB
```

Observation 5: Create a new column awaiting_time_days

In [11]:
```
# Create awaiting_time_days column
df['awaiting_time_days'] = (df.appointment_day - df.scheduled_day).dt.days # and conver
```

```
# Check if the column exists
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 15 columns):
patient_id            110527 non-null int64
appointment_id        110527 non-null int64
gender                110527 non-null object
scheduled_day         110527 non-null datetime64[ns]
appointment_day       110527 non-null datetime64[ns]
age                   110527 non-null int64
neighbourhood         110527 non-null object
scholarship           110527 non-null int64
hypertension          110527 non-null int64
diabetes              110527 non-null int64
alcoholism            110527 non-null int64
handicap              110527 non-null int64
sms_received          110527 non-null int64
no_show               110527 non-null object
awaiting_time_days    110527 non-null int64
dtypes: datetime64[ns](2), int64(10), object(3)
memory usage: 12.6+ MB
```

Observation 6: Create a new column appointment_dow (day of week appointment)

```
In [12]: # Create appointment_dow column
         df['appointment_dow'] = df.scheduled_day.dt.weekday_name

         # Check the values
         df['appointment_dow'].value_counts()
```

```
Out[12]: Tuesday      26168
         Wednesday    24262
         Monday       23085
         Friday       18915
         Thursday     18073
         Saturday        24
         Name: appointment_dow, dtype: int64
```

The distribution of appointments among days of week (Monday-Friday) is almost equal with a little bit less visits on Thursday and Friday. There are 24 visits on Saturday and none on Sunday.

```
In [13]: df.describe()
```

```
Out[13]:          patient_id  appointment_id            age    scholarship  \
         count  1.105270e+05    1.105270e+05  110527.000000  110527.000000
```

```
mean   1.474963e+14   5.675305e+06     37.088874         0.098266
std    2.560949e+14   7.129575e+04     23.110205         0.297675
min    3.921700e+04   5.030230e+06     -1.000000         0.000000
25%    4.172614e+12   5.640286e+06     18.000000         0.000000
50%    3.173184e+13   5.680573e+06     37.000000         0.000000
75%    9.439172e+13   5.725524e+06     55.000000         0.000000
max    9.999816e+14   5.790484e+06    115.000000         1.000000

        hypertension       diabetes    alcoholism       handicap  \
count  110527.000000  110527.000000  110527.000000  110527.000000
mean        0.197246       0.071865       0.030400       0.022248
std         0.397921       0.258265       0.171686       0.161543
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       4.000000

        sms_received  awaiting_time_days
count  110527.000000       110527.000000
mean        0.321026           10.183702
std         0.466873           15.254996
min         0.000000           -6.000000
25%         0.000000            0.000000
50%         0.000000            4.000000
75%         1.000000           15.000000
max         1.000000          179.000000
```

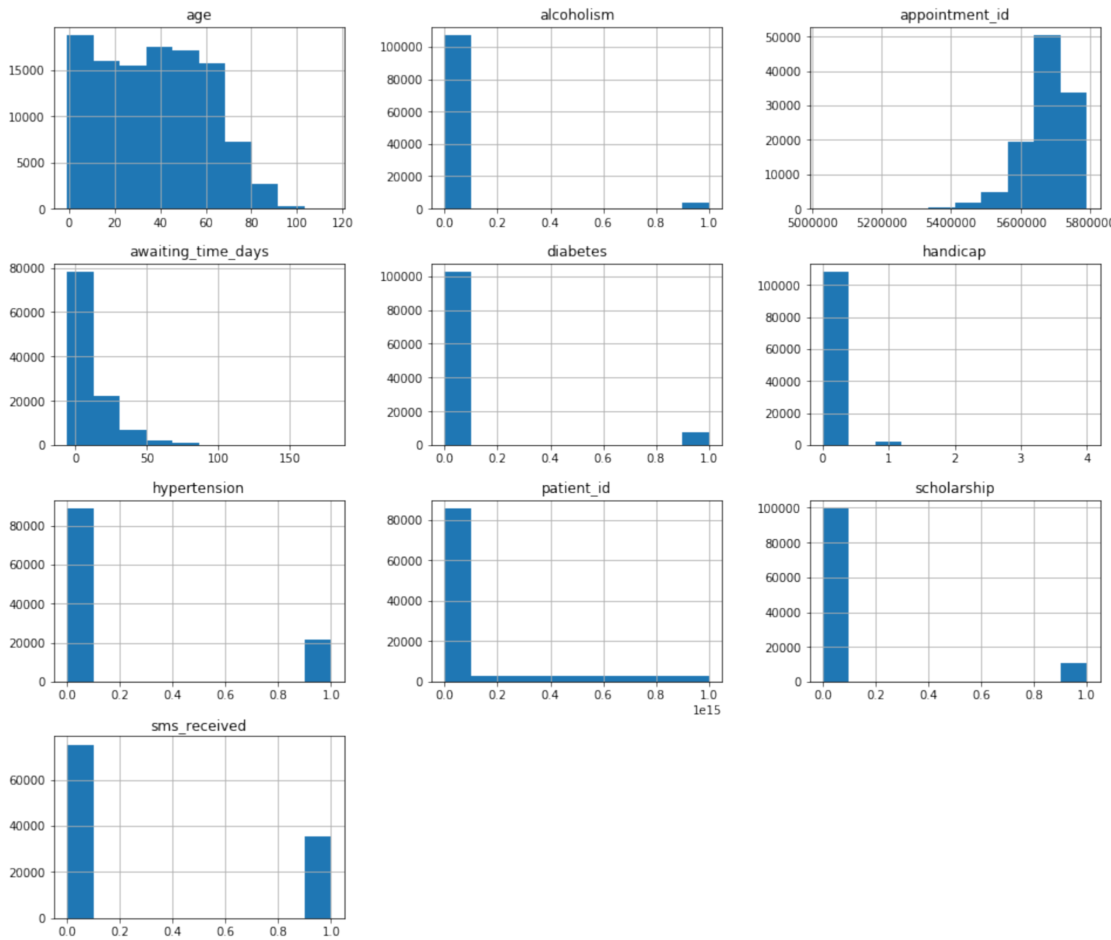
Additional observations

1)**age***: The patients are 37 years on average. 25% of patients are below 18 and most of them are below 55. There is a data range problem in the dataset. The age value cannot be below 0, and there are some very old people as well. To be investigated.

2)***handicap***: is represented by 4 classes as opposed to other categorical variables in this dataset. This can be a result of an error or there are 4 categories used. Both options are potentially valid and this should be confirmed by an SME. sms_received: 75% of patients received sms regarding an appointment.

3)***awaiting_time_days:*** 10 days on average patients waited for an appointment. 50% of patients waited up to 4 days and 75% up to 15 days for an appointment. The longest awaiting time was 179 days. There is at least one case where a visit happened 6 days before it was scheduled. This should not happen and will be further investigated.

```
In [14]: df.hist(figsize=(16,14));
```



Histogram observations

1)age: There are many very young people in the dataset but in general the patients age is distributed evenly and the number of patients goes drastricly down for patients older than 60 years.

2)alcoholism: Most of the patients are not alcoholics.

3)diabetes: Most of the patients are not diabetes but more than alcoholics.

4)handicap: There are for handicap categories with most of the people not being handicapted.

5)hypertension: Most patients do not have hypertension diagnosed.

gender

```
In [15]: # Print Unique Values
         print("Unique Values in `gender` => {}".format(df.gender.unique()))
```

```
Unique Values in `gender` => ['F' 'M']
```

scheduled_day

```
In [16]: # Print Unique Values
         print("Unique Values in `scheduled_day` => {}".format(df.scheduled_day.unique()))
```

```
Unique Values in `scheduled_day` => ['2016-04-29T00:00:00.000000000' '2016-04-27T00:00:00.000000
 '2016-04-26T00:00:00.000000000' '2016-04-28T00:00:00.000000000'
 '2016-04-25T00:00:00.000000000' '2016-04-20T00:00:00.000000000'
 '2016-03-31T00:00:00.000000000' '2016-04-19T00:00:00.000000000'
 '2016-04-06T00:00:00.000000000' '2016-04-18T00:00:00.000000000'
 '2016-04-11T00:00:00.000000000' '2016-04-12T00:00:00.000000000'
 '2016-04-15T00:00:00.000000000' '2016-04-01T00:00:00.000000000'
 '2016-04-05T00:00:00.000000000' '2016-04-08T00:00:00.000000000'
 '2016-04-14T00:00:00.000000000' '2016-04-13T00:00:00.000000000'
 '2016-04-07T00:00:00.000000000' '2016-03-17T00:00:00.000000000'
 '2016-03-30T00:00:00.000000000' '2016-03-29T00:00:00.000000000'
 '2016-03-18T00:00:00.000000000' '2016-03-28T00:00:00.000000000'
 '2016-03-04T00:00:00.000000000' '2016-03-15T00:00:00.000000000'
 '2016-03-14T00:00:00.000000000' '2016-03-21T00:00:00.000000000'
 '2016-03-23T00:00:00.000000000' '2016-03-22T00:00:00.000000000'
 '2016-03-16T00:00:00.000000000' '2016-03-10T00:00:00.000000000'
 '2016-02-29T00:00:00.000000000' '2016-03-08T00:00:00.000000000'
 '2016-03-07T00:00:00.000000000' '2016-02-24T00:00:00.000000000'
 '2016-02-22T00:00:00.000000000' '2016-01-29T00:00:00.000000000'
 '2016-02-23T00:00:00.000000000' '2016-02-05T00:00:00.000000000'
 '2016-02-11T00:00:00.000000000' '2016-02-02T00:00:00.000000000'
 '2016-01-05T00:00:00.000000000' '2016-01-11T00:00:00.000000000'
 '2016-02-26T00:00:00.000000000' '2016-02-19T00:00:00.000000000'
 '2016-02-17T00:00:00.000000000' '2016-03-03T00:00:00.000000000'
 '2016-03-02T00:00:00.000000000' '2016-03-09T00:00:00.000000000'
 '2016-03-01T00:00:00.000000000' '2016-03-19T00:00:00.000000000'
 '2016-03-11T00:00:00.000000000' '2016-02-16T00:00:00.000000000'
 '2016-02-25T00:00:00.000000000' '2016-04-09T00:00:00.000000000'
 '2016-05-24T00:00:00.000000000' '2016-05-25T00:00:00.000000000'
 '2016-05-31T00:00:00.000000000' '2016-05-17T00:00:00.000000000'
 '2016-05-30T00:00:00.000000000' '2016-05-12T00:00:00.000000000'
 '2016-05-19T00:00:00.000000000' '2016-05-10T00:00:00.000000000'
 '2016-05-02T00:00:00.000000000' '2016-05-16T00:00:00.000000000'
 '2016-05-04T00:00:00.000000000' '2016-05-13T00:00:00.000000000'
 '2016-05-20T00:00:00.000000000' '2016-05-05T00:00:00.000000000'
 '2016-05-18T00:00:00.000000000' '2016-05-06T00:00:00.000000000'
 '2016-05-09T00:00:00.000000000' '2016-05-03T00:00:00.000000000'
 '2016-05-11T00:00:00.000000000' '2015-11-10T00:00:00.000000000'
 '2016-02-18T00:00:00.000000000' '2016-02-03T00:00:00.000000000'
 '2016-01-14T00:00:00.000000000' '2016-01-21T00:00:00.000000000'
 '2016-01-28T00:00:00.000000000' '2016-02-01T00:00:00.000000000'
 '2015-12-14T00:00:00.000000000' '2015-12-08T00:00:00.000000000'
 '2016-01-07T00:00:00.000000000' '2016-04-30T00:00:00.000000000'
 '2016-04-16T00:00:00.000000000' '2016-02-04T00:00:00.000000000'
 '2015-12-03T00:00:00.000000000' '2016-01-04T00:00:00.000000000'
 '2016-01-13T00:00:00.000000000' '2016-02-12T00:00:00.000000000'
 '2016-01-20T00:00:00.000000000' '2016-01-22T00:00:00.000000000'
 '2016-01-25T00:00:00.000000000' '2016-01-27T00:00:00.000000000'
```

```
 '2016-01-19T00:00:00.000000000' '2016-02-15T00:00:00.000000000'
 '2016-05-14T00:00:00.000000000' '2016-05-07T00:00:00.000000000'
 '2016-06-02T00:00:00.000000000' '2016-06-03T00:00:00.000000000'
 '2016-06-01T00:00:00.000000000' '2016-06-06T00:00:00.000000000'
 '2016-06-07T00:00:00.000000000' '2016-06-08T00:00:00.000000000'
 '2016-06-04T00:00:00.000000000' '2016-01-26T00:00:00.000000000'
 '2015-12-07T00:00:00.000000000' '2015-12-15T00:00:00.000000000'
 '2016-03-05T00:00:00.000000000']
```

appointment_day

```
In [17]: # Print Unique Values
         print("Unique Values in `appointment_day` => {}".format(df.appointment_day.unique()))
```

```
Unique Values in `appointment_day` => ['2016-04-29T00:00:00.000000000' '2016-05-03T00:00:00.0000
 '2016-05-10T00:00:00.000000000' '2016-05-17T00:00:00.000000000'
 '2016-05-24T00:00:00.000000000' '2016-05-31T00:00:00.000000000'
 '2016-05-02T00:00:00.000000000' '2016-05-30T00:00:00.000000000'
 '2016-05-16T00:00:00.000000000' '2016-05-04T00:00:00.000000000'
 '2016-05-19T00:00:00.000000000' '2016-05-12T00:00:00.000000000'
 '2016-05-06T00:00:00.000000000' '2016-05-20T00:00:00.000000000'
 '2016-05-05T00:00:00.000000000' '2016-05-13T00:00:00.000000000'
 '2016-05-09T00:00:00.000000000' '2016-05-25T00:00:00.000000000'
 '2016-05-11T00:00:00.000000000' '2016-05-18T00:00:00.000000000'
 '2016-05-14T00:00:00.000000000' '2016-06-02T00:00:00.000000000'
 '2016-06-03T00:00:00.000000000' '2016-06-06T00:00:00.000000000'
 '2016-06-07T00:00:00.000000000' '2016-06-01T00:00:00.000000000'
 '2016-06-08T00:00:00.000000000']
```

age

```
In [18]: # Print Unique Values
         print("Unique Values in `age` => {}".format(df.age.unique()))
```

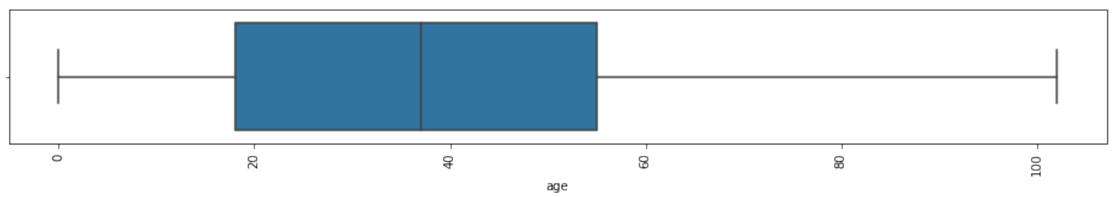
```
Unique Values in `age` => [ 62  56   8  76  23  39  21  19  30  29  22  28  54  15  50  40  46
  13  65  45  51  32  12  61  38  79  18  63  64  85  59  55  71  49  78
  31  58  27   6   2  11   7   0   3   1  69  68  60  67  36  10  35  20
  26  34  33  16  42   5  47  17  41  44  37  24  66  77  81  70  53  75
  73  52  74  43  89  57  14   9  48  83  72  25  80  87  88  84  82  90
  94  86  91  98  92  96  93  95  97 102 115 100  99  -1]
```

```
In [19]: print('Before change')
         print("Patients with `Age` less than -1 -> {}".format(df[df.age == -1].shape[0]))
         print("Patients with `Age` equal to 0 -> {}".format(df[df.age == 0].shape[0]))
         print("Patients with `Age` greater than 110 -> {}".format(df[df.age > 110].shape[0]))
```

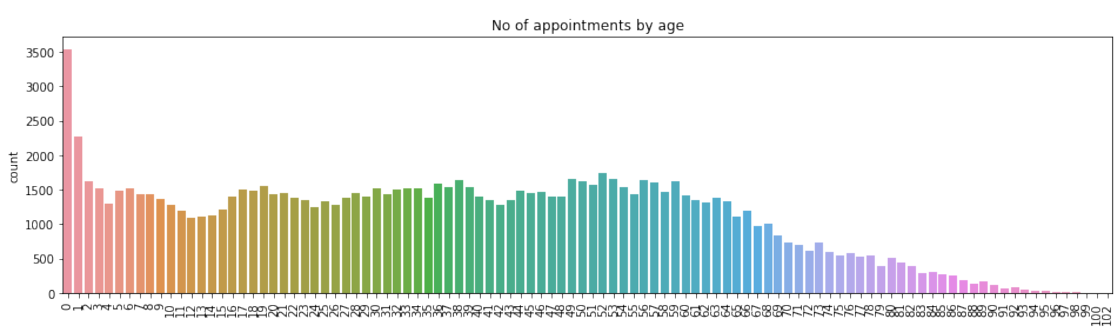
```
df = df[(df.age >= 0) & (df.age <= 110)]
df.age.value_counts()

print('After change')
print("Patients with `Age` less than -1 -> {}".format(df[df.age == -1].shape[0]))
print("Patients with `Age` equal to 0 -> {}".format(df[df.age == 0].shape[0]))
print("Patients with `Age` greater than 110 -> {}".format(df[df.age > 110].shape[0]))
```

```
Before change
Patients with `Age` less than -1 -> 1
Patients with `Age` equal to 0 -> 3539
Patients with `Age` greater than 110 -> 5
After change
Patients with `Age` less than -1 -> 0
Patients with `Age` equal to 0 -> 3539
Patients with `Age` greater than 110 -> 0
```

In [20]:
```
# Let's see a boxplot showing what is age values distribution (already seen above in a
plt.figure(figsize=(16,2))
plt.xticks(rotation=90)
_ = sns.boxplot(x=df.age)
```



In [21]:
```
# Let's see how many there are patients of each age
plt.figure(figsize=(16,4))
plt.xticks(rotation=90)
ax = sns.countplot(x=df.age)
ax.set_title("No of appointments by age")
plt.show()
```



scholarship

```
In [22]: # Print Unique Values
         print("Unique Values in `scholarship` => {}".format(df.scholarship.unique()))
```

```
Unique Values in `scholarship` => [0 1]
```

hypertension

```
In [23]: # Print Unique Values
         print("Unique Values in `hypertension` => {}".format(df.hypertension.unique()))
```

```
Unique Values in `hypertension` => [1 0]
```

diabetes

```
In [24]: # Print Unique Values
         print("Unique Values in `diabetes` => {}".format(df.diabetes.unique()))
```

```
Unique Values in `diabetes` => [0 1]
```

alcoholism

```
In [25]: # Print Unique Values
         print("Unique Values in `alcoholism` => {}".format(df.alcoholism.unique()))
```

```
Unique Values in `alcoholism` => [0 1]
```

handicap

```
In [26]: # Print Unique Values
         print("Unique Values in `handicap` => {}".format(df.handicap.unique()))
```

```
Unique Values in `handicap` => [0 1 2 3 4]
```

```
In [27]: # The handicap column contains 4 numeric values (classes), which is unusual comparing t
         df.handicap.value_counts()
```

```
Out[27]: 0    108284
         1      2038
         2       183
         3        13
         4         3
         Name: handicap, dtype: int64
```

sms_received

```
In [28]: # Print Unique Values
         print("Unique Values in `sms_received` => {}".format(df.sms_received.unique()))
```

```
Unique Values in `sms_received` => [0 1]
```

awaiting_time_days

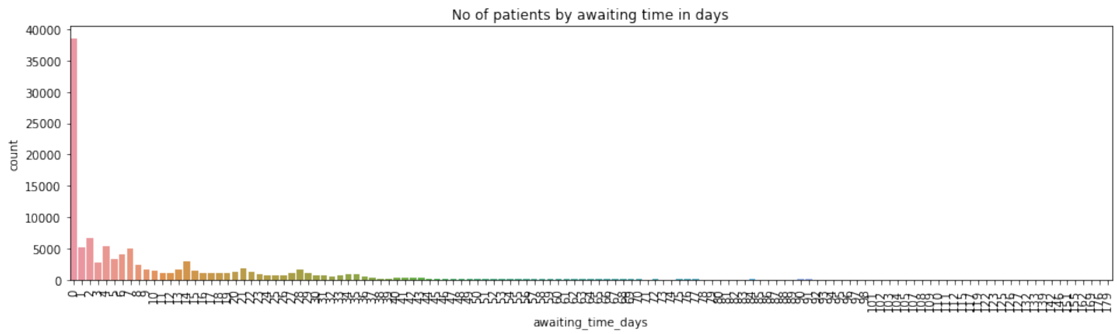
```
In [29]: # Print Unique Values
         print("Unique Values in `awaiting_time_days` => {}".format(df.awaiting_time_days.unique
```

```
Unique Values in `awaiting_time_days` => [  0   2   3   1   4   9  29  10  23  11  18  17  14  2
  22  43  30  31  42  32  56  45  46  39  37  38  44  50  60  52  53  65
  67  91  66  84  78  87 115 109  63  70  72  57  58  51  59  41  49  73
  64  20  33  34   6  35  36  12  13  40  47   8   5   7  25  26  48  27
  19  61  55  62 176  54  77  69  83  76  89  81 103  79  68  75  85 112
  -1  80  86  98  94 142 155 162 169 104 133 125  96  88  90 151 126 127
 111 119  74  71  82 108 110 102 122 101 105  92  97  93 107  95  -6 139
 132 179 117 146 123]
```

```
In [30]: # Awaiting time cannot be less than 0. I am assuming that a visit cannot happen before
         # Let's see how many such values exist
         print('Before change: {}'.format(df[(df.awaiting_time_days < 0)].awaiting_time_days.val

         # I will remove all records with such values.
         df = df[(df.awaiting_time_days >= 0)]

         #Check if any awaiting time days values below 0 left in the dataset
         print('After change: {}'.format(df[(df.awaiting_time_days < 0)].awaiting_time_days.valu
```

```
Before change: -1    4
-6    1
Name: awaiting_time_days, dtype: int64
After change: Series([], Name: awaiting_time_days, dtype: int64)
```

```
In [31]: # Let's see how many there are patients of each age
         plt.figure(figsize=(16,4))
         plt.xticks(rotation=90)
         ax = sns.countplot(x=df.awaiting_time_days)
         ax.set_title("No of patients by awaiting time in days")
         plt.show()
```



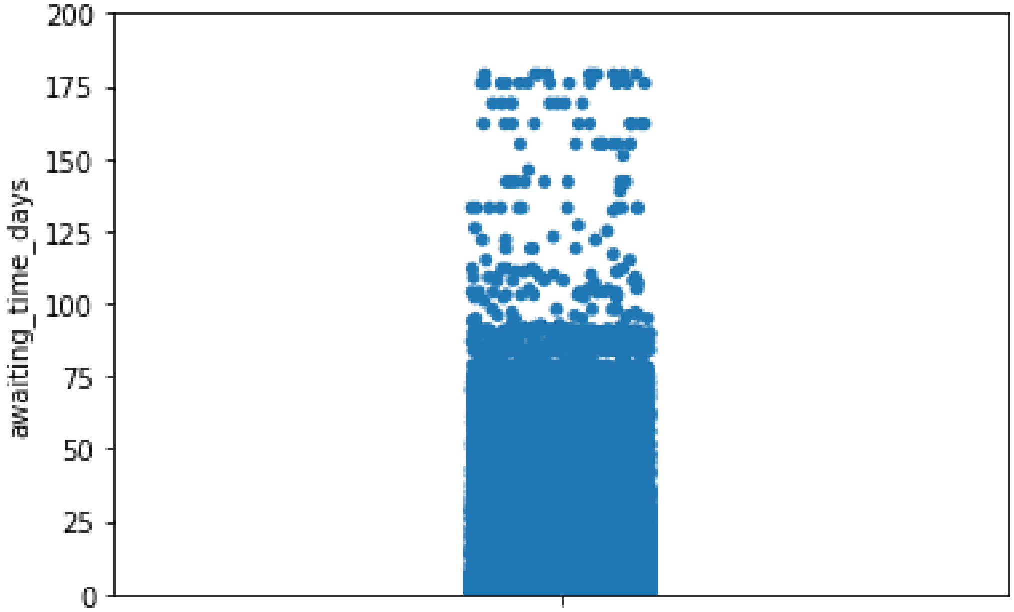

```
In [32]: # Return number of patients with awaiting_time_days == 0
         awaiting0 = df[(df.awaiting_time_days == 0)].awaiting_time_days.value_counts()
         awaiting0
```

```
Out[32]: 0    38561
         Name: awaiting_time_days, dtype: int64
```

```
In [33]: awaiting0_not_showed_up = len(df.query('awaiting_time_days  == 0 and no_show == "Yes"')
         awaiting0_not_showed_up_ratio = int(round(awaiting0_not_showed_up/awaiting0[0]*100))
         print('Out of all patients scheduling an appointment for the same day (in total {}), {}
```

```
Out of all patients scheduling an appointment for the same day (in total 38561), 1792 of patient
```

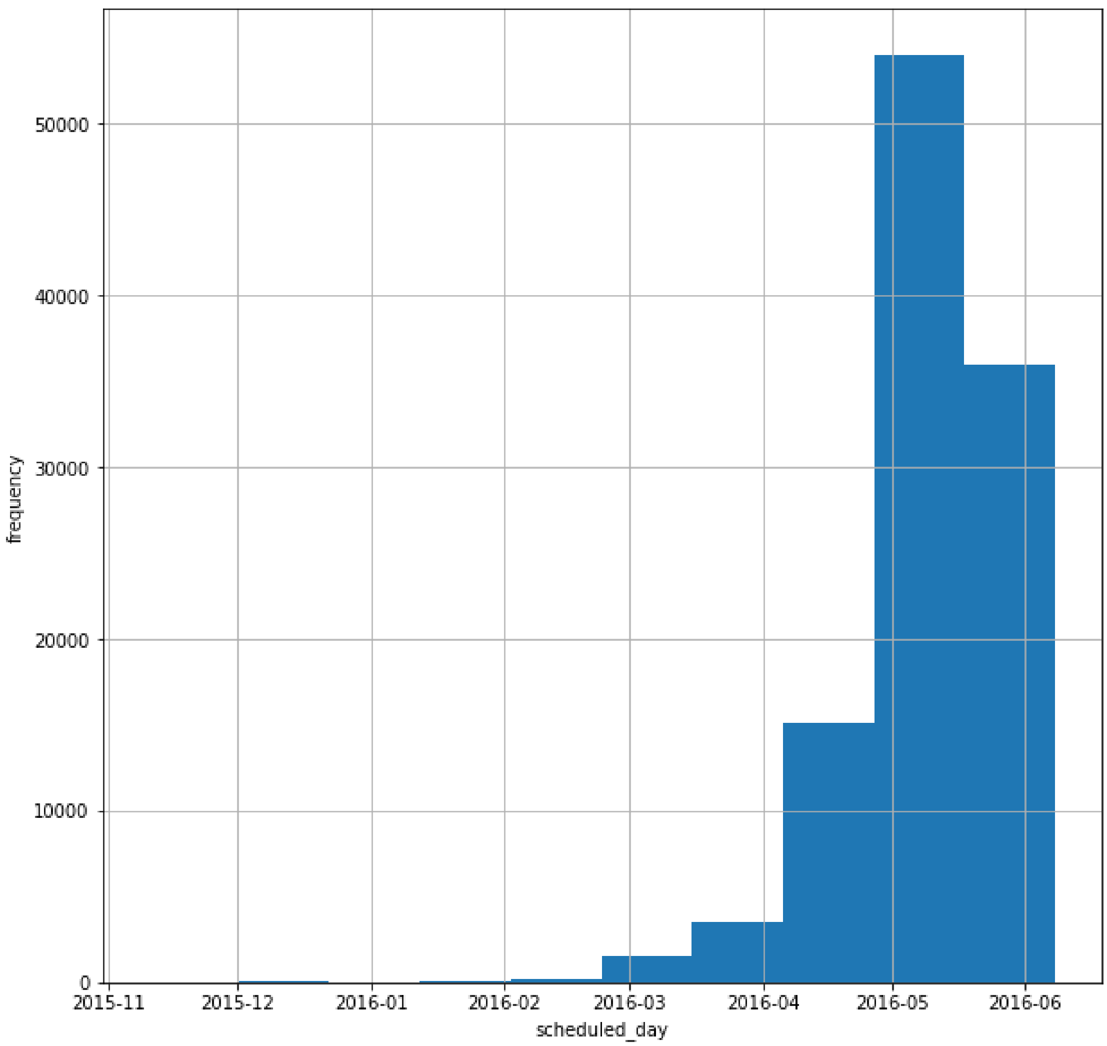
```
In [34]: # It seems that most of the visits happened within 3 months from being scheduled
         sns.stripplot(data = df, y = 'awaiting_time_days', jitter = True)
         plt.ylim(0, 200)
         plt.show();
```



scheduled_day

```
In [35]: print('Scheduling visits started on: {}.'.format(df['scheduled_day'].min()))
         print('Scheduling visits ended on: {}.'.format(df['scheduled_day'].max()))

         fig = plt.figure(figsize=(10, 10))
         ax = fig.add_subplot(1, 1, 1)
         ax.set_xlabel('scheduled_day')
         ax.set_ylabel('frequency')
         df['scheduled_day'].hist();
```

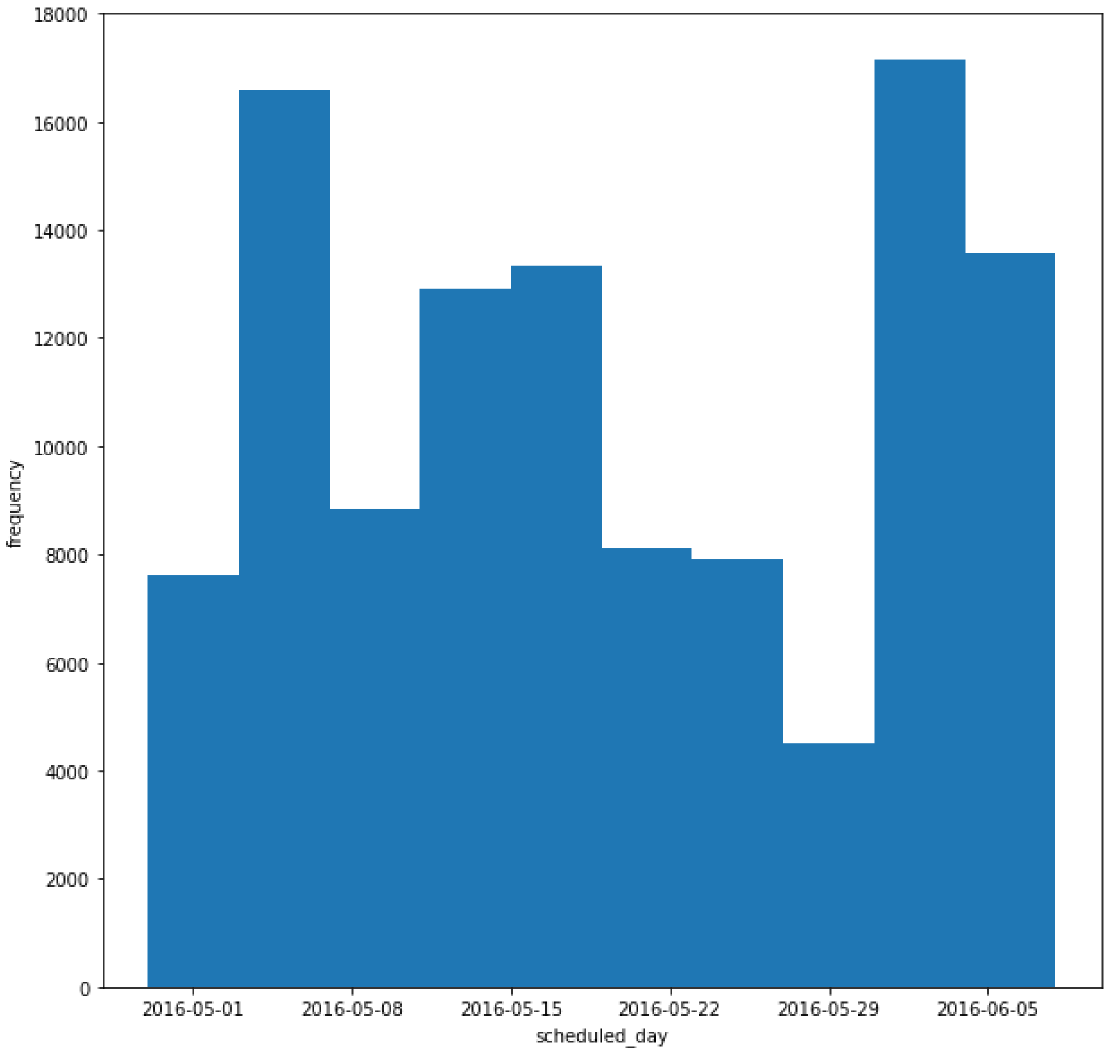
```
Scheduling visits started on: 2015-11-10 00:00:00.
Scheduling visits ended on: 2016-06-08 00:00:00.
```



appointment_day

```
In [36]: print('Visit appointments started on: {}.'.format(df['appointment_day'].min()))
         print('Visit appointments ended on: {}.'.format(df['appointment_day'].max()))

         fig = plt.figure(figsize=(10, 10))
         ax = fig.add_subplot(1,1,1)
         ax.set_xlabel('scheduled_day')
         ax.set_ylabel('frequency')
         df['appointment_day'].hist(grid=False, ax=ax);
```

```
Visit appointments started on: 2016-04-29 00:00:00.
Visit appointments ended on: 2016-06-08 00:00:00.
```



appointment_dow

```
In [37]: # Print Unique Values
         print("Unique Values in `appointment_dow` => {}".format(df.appointment_dow.unique()))
```

```
Unique Values in `appointment_dow` => ['Friday' 'Wednesday' 'Tuesday' 'Thursday' 'Monday' 'Satur
```

appointment_id

```
In [38]: # Are the appointments ids unique?
         # If yes, then num_unique_apps will be equal to number of all records in our dataset
         num_unique_apps = len(df.appointment_id.unique())
         all_dataset_rec_number = df.shape[0]
         print('{} == {}'.format(num_unique_apps, all_dataset_rec_number))
```

```
110516 == 110516
```

Questions Based on dataset analysis, I will focus on putting more light on answers to the following questions:

1)How many percent of patients missed their scheduled appointment?

2)What is the gender distribution for show / no-show patients?

3)Are there patients with more than one appointment? If yes, what are the top 10 patients with most appointments?

4)What factors are important to know in order to predict if a patient will show up for their scheduled appointment?

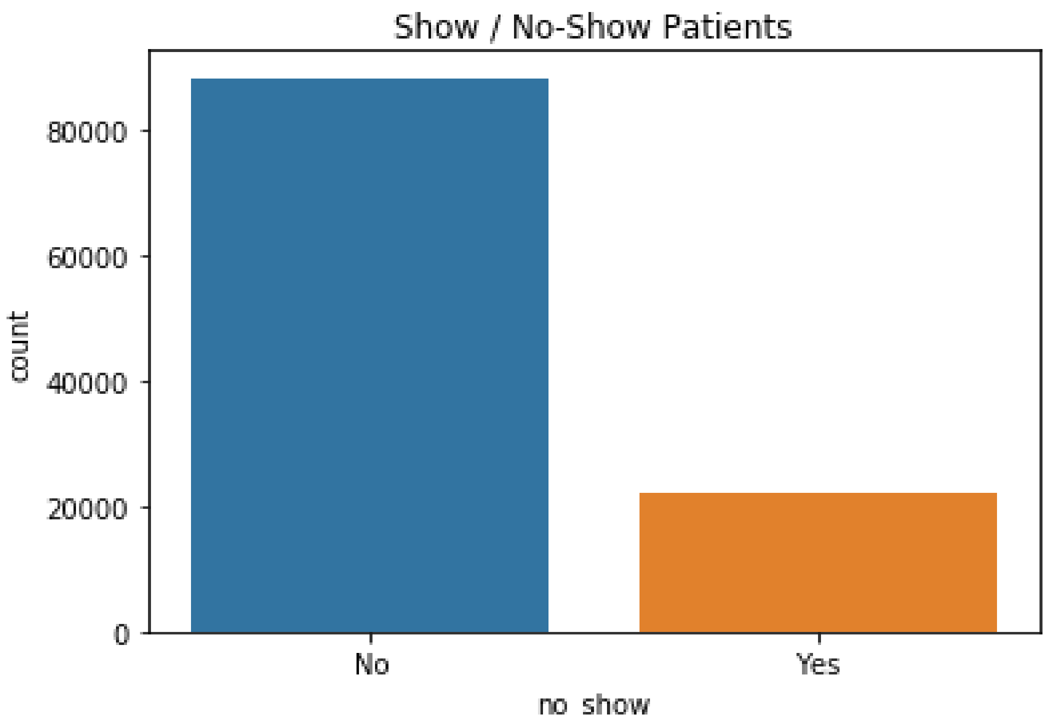
5)What is age distribution of diabetes who showed and did not show up?

6)How activities done by an appointment scheduling office (sending SMS, participation in scholarship) influence show / no-show ratio?

2 Exploratory Data Analysis

1. How many percent of patients missed their scheduled appointment?

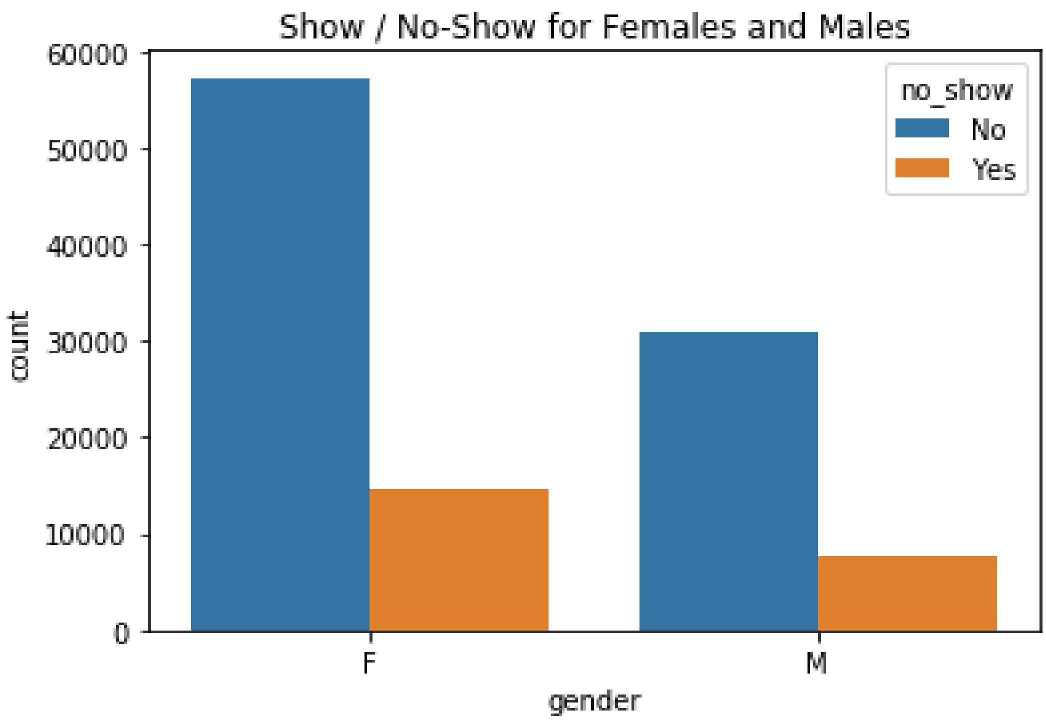
```
In [39]: all_appointments = df.shape[0]
         missed_appointments = len(df.query('no_show == \'Yes\''))
         missed_ratio = int(round(missed_appointments/all_appointments*100))

         ax = sns.countplot(x=df.no_show, data=df)
         ax.set_title("Show / No-Show Patients")
         plt.show();

         print('{}% of appointments were missed.'.format(missed_ratio))
```




```
20% of appointments were missed.
```

2. What is the gender distribution for show / no-show patients?

```
In [40]: all_appointments_by_f = len(df.loc[df['gender'] == "F"])
         all_appointments_by_m = len(df.loc[df['gender'] == "M"])

         missed_appointments_by_f = len(df.query('no_show == "Yes" and gender == "F"'))
         missed_appointments_by_m = len(df.loc[(df['gender'] == "M") & (df['no_show'] == "Yes")]

         missed_ratio_f = int(round(missed_appointments_by_f/all_appointments_by_f*100))
         missed_ratio_m = int(round(missed_appointments_by_m/all_appointments_by_m*100))

         ax = sns.countplot(x=df.gender, hue=df.no_show, data=df)
         ax.set_title("Show / No-Show for Females and Males")
         x_ticks_labels=['Female', 'Male']
         plt.show();

         print('Out of {} appointments made by females, {} were missed with the ratio of {}%.'.f
         print('Out of {} appointments made by males, {} were missed with the ratio of {}%.'.for
```




```
Out of 71831 appointments made by females, 14588 were missed with the ratio of 20%.
Out of 38685 appointments made by males, 7723 were missed with the ratio of 20%.
```

3. Are there patients with more than one appointment? If yes, what are the top 10 patients with most appointments?

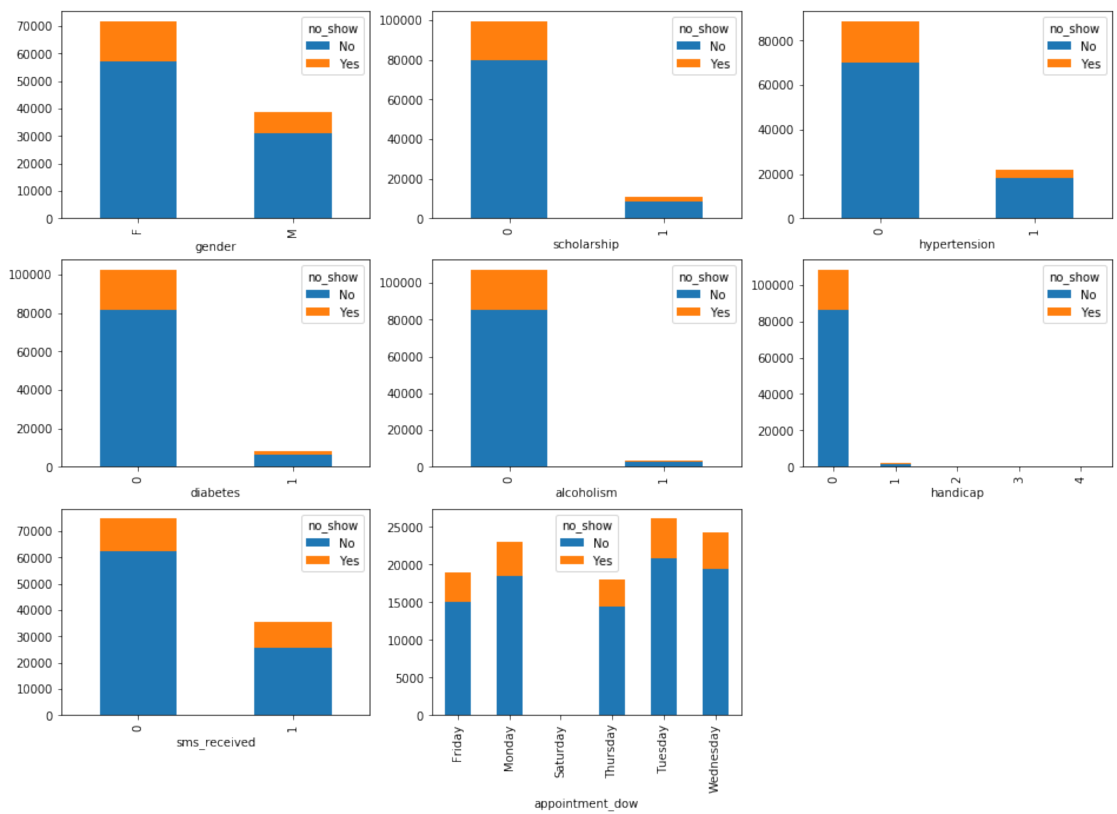
```
In [41]: df.patient_id.value_counts().iloc[0:10]
```

```
Out[41]: 822145925426128    88
         99637671331        84
         26886125921145     70
         33534783483176     65
         258424392677       62
         75797461494159     62
         871374938638855    62
         6264198675331      62
         66844879846766     57
         872278549442       55
         Name: patient_id, dtype: int64
```

There are patients with multiple appointments. The number of appointments of top 10 patients range from 88 to 55. Taking into consideration, that the time range of visits appointed spans over 1.5 months, an appointment is most likely each examination or each specialist visit. So within one patient visit in a hospital, there could be multiple appointments scheduled. One of the no-show reasons could be the fact, that patients could be too tired to take part in all examinations during a particular visit, or the open hours were not sufficient to show up in all appointments. There could be also other reasons. The high number of appointments over so short period of time should be consulted with an SME to decide if performing (or not) additional analysis in this area makes sense.

4. What factors are important to know in order to predict if a patient will show up for their scheduled appointment?

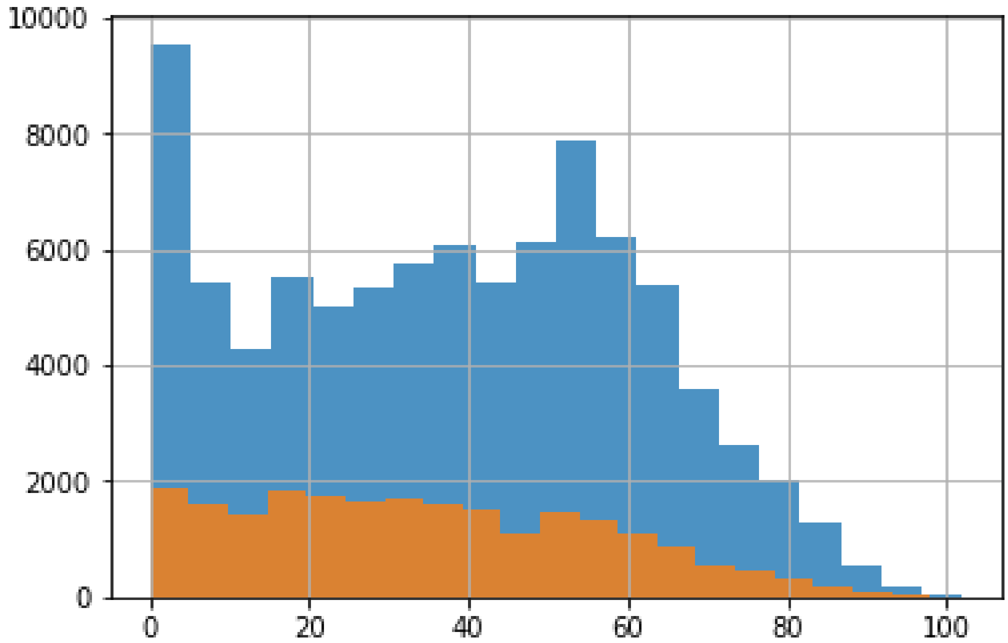
In [42]:
```
# First, let's look at categorical variables
categorical_vars = ['gender', 'scholarship', 'hypertension', 'diabetes', 'alcoholism',

fig = plt.figure(figsize=(16, 11))
for i, var in enumerate(categorical_vars):
    ax = fig.add_subplot(3, 3, i+1)
    df.groupby([var, 'no_show'])[var].count().unstack('no_show').plot(ax=ax, kind='bar'
```



For all categorical variables the distributions of show / no-show for different categories look very similar. There is no clear indication of any of these variables having bigger then others impact on show / no-show characteristics. The charts confirm about 20% no-show rate for most categories.

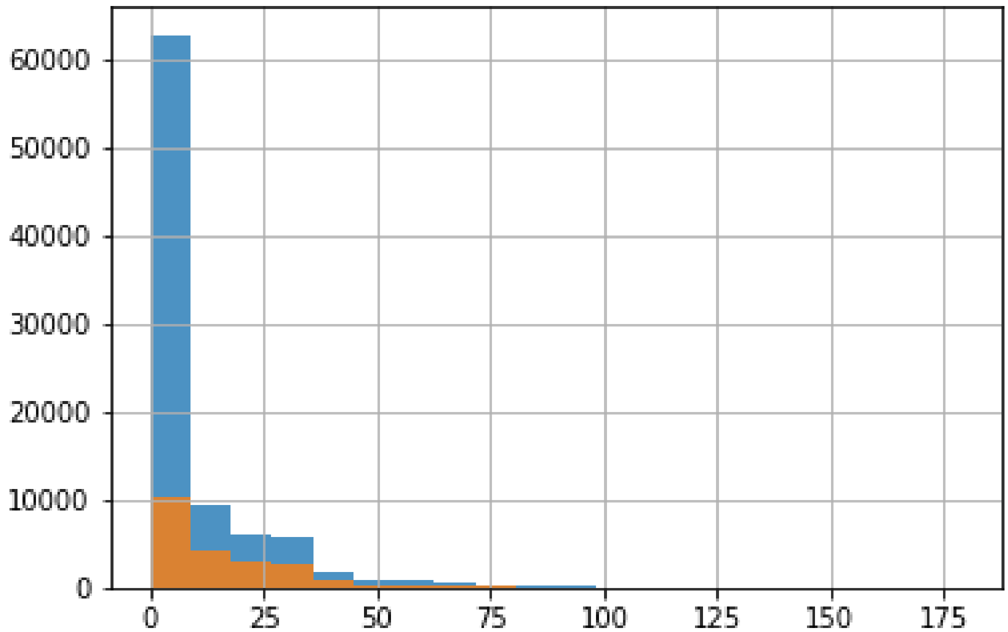
In [43]:
```
# Two useful masks to be used in further analysis
showed = df.no_show == 'No'
not_showed = df.no_show == 'Yes'
```

In [44]:
```
# Let's now look closer to numerical variables
# Age:
df.age[showed].hist(alpha=0.8, bins=20);
df.age[not_showed].hist(alpha=0.8, bins=20);
```



Infants show up most often then people around 50-55. But between 1-65 years old, the rate of no-shows seems to be higher than 20%.

In [45]:
```
# Number of days between the date of scheduling an appointment and the appointment itse
df.awaiting_time_days[showed].hist(alpha=0.8, bins=20);
df.awaiting_time_days[not_showed].hist(alpha=0.8, bins=20);
```

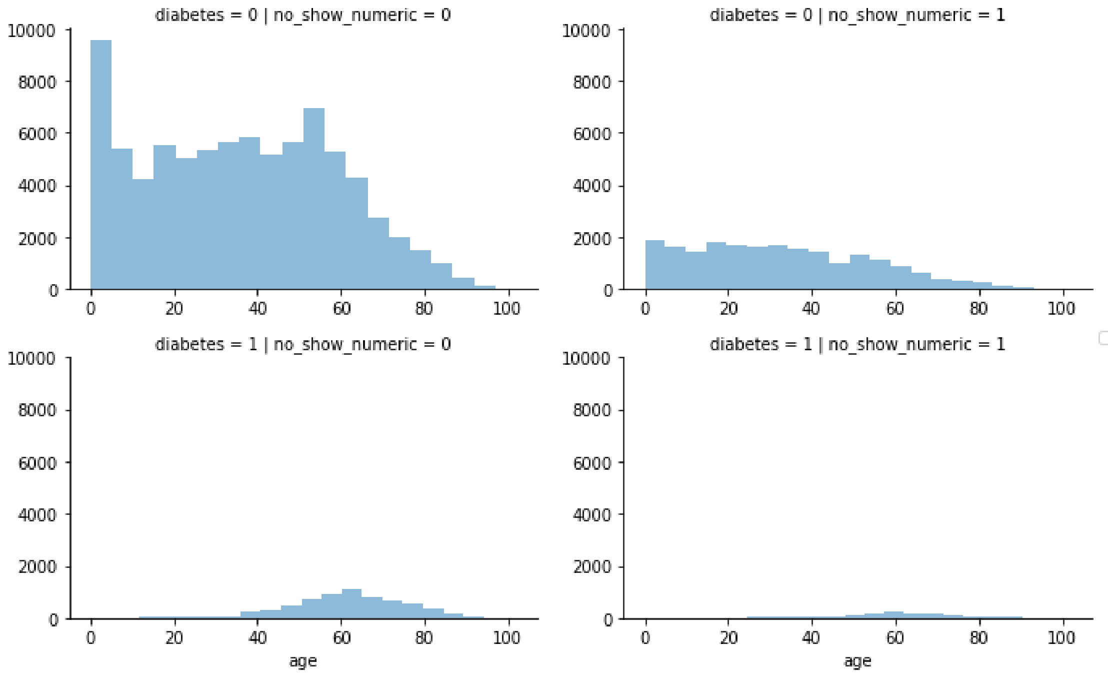


5. What is age distribution of diabetes who showed and did not show up?

```
In [46]: # This is a helper column representing no_shows in a numerical form (Yes->1, No->0)
         df['no_show_numeric'] = np.where(df['no_show']=='Yes', 1, 0)

In [47]: df[['diabetes', 'no_show_numeric']].groupby(['diabetes'], as_index=False).mean().sort_v

Out[47]:    diabetes  no_show_numeric
         0         0         0.203572
         1         1         0.180033
```

In general, 18% of diabetes did not show up, which is about 2% lower from the general average of not showing up. It seems that diabetes are more careful about their health and take medial appointments more seriously then non diabetes.

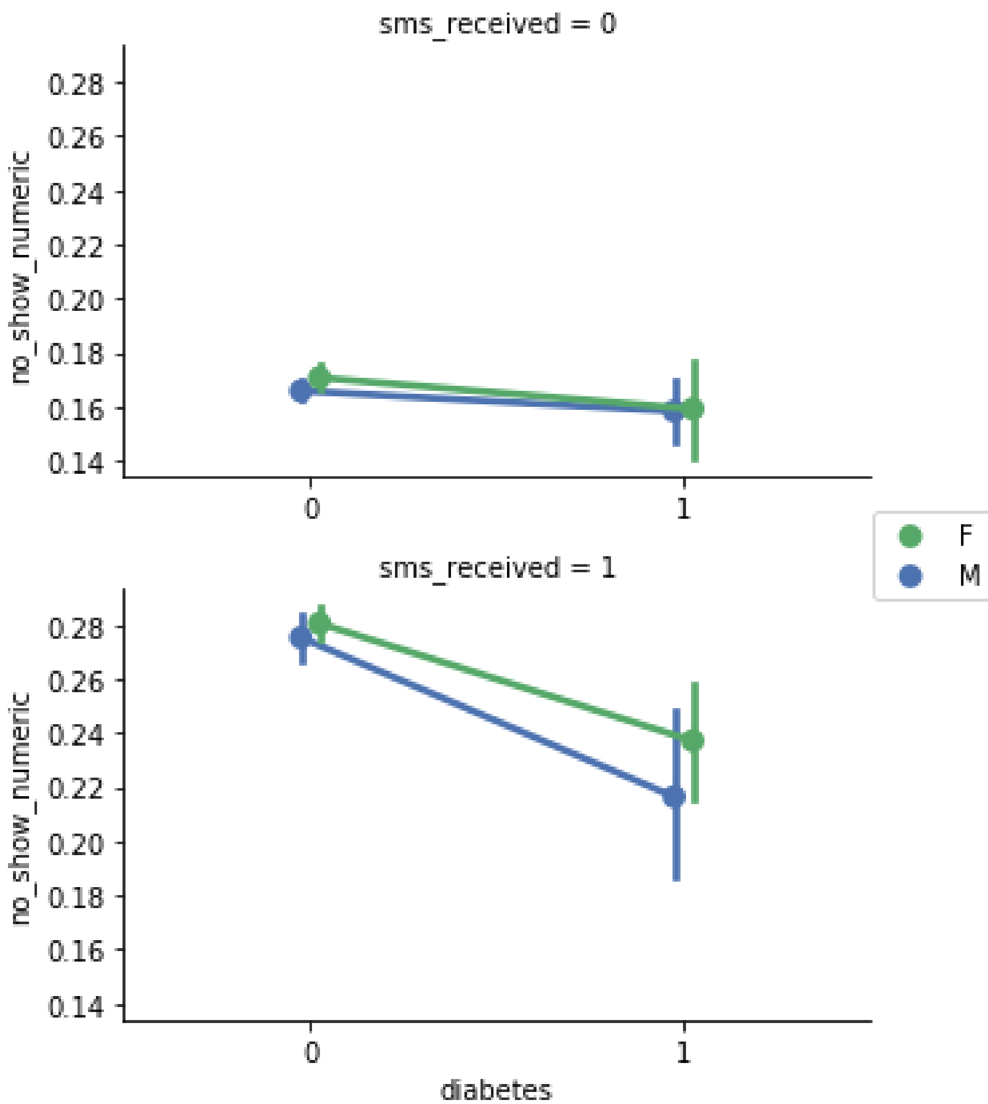
```
In [48]: grid = sns.FacetGrid(df, col='no_show_numeric', row='diabetes', aspect=1.6)
         grid.map(plt.hist, 'age', alpha=.5, bins=20)
         grid.add_legend();
```



The diabetes distribution shape is symetrical. The mean of this distribution is about 60. To calculate it more precisely, as well as its standard deviation a statistical method should be used.

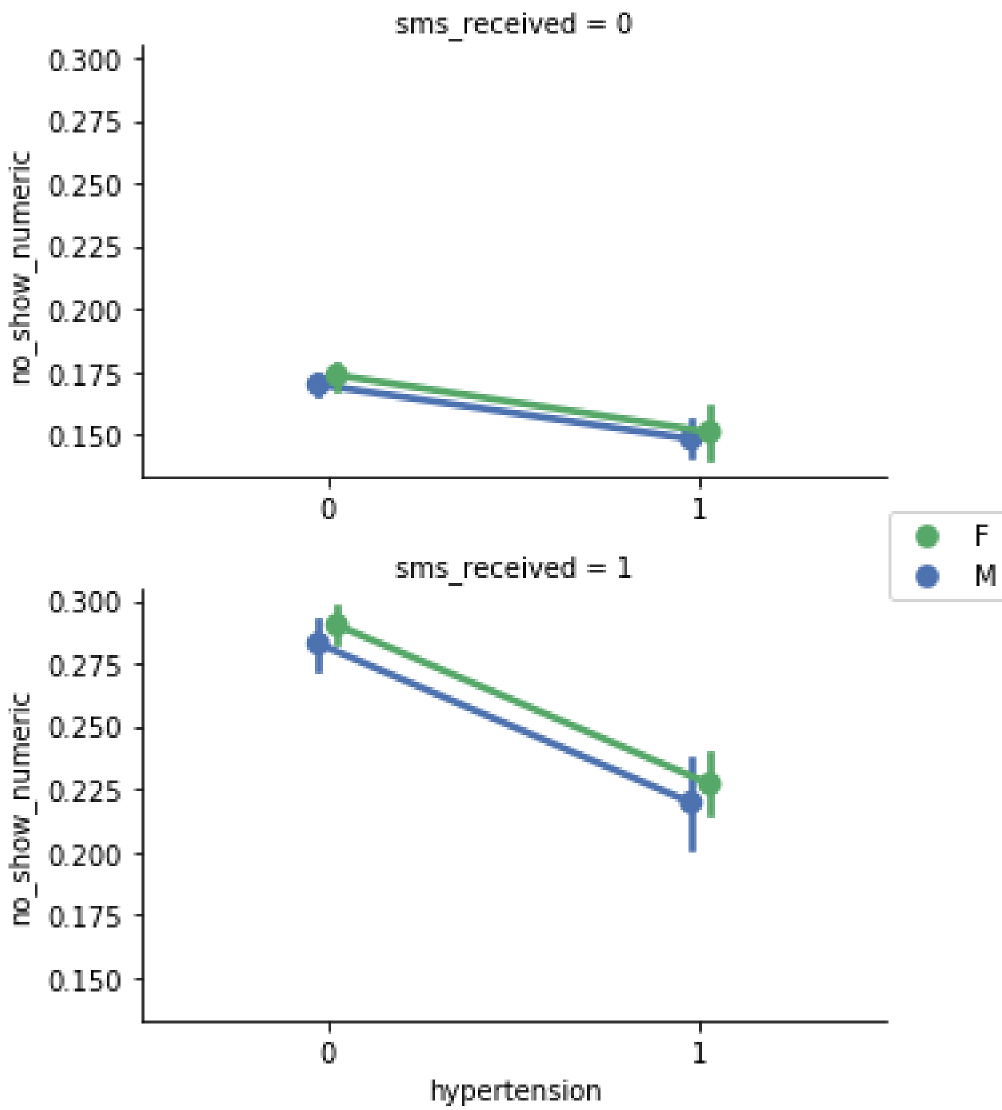
6. How activities done by an appointment scheduling office (sending SMS) influence show / no-show ratio?

In [49]:
```
# The Pointplot uses bootstraping method to estimate of a mean and a std error
# Appointments related to patients with diabetes and receiving SMS:
grid = sns.FacetGrid(df, row='sms_received', aspect=1.6)
grid.map(sns.pointplot, 'diabetes', 'no_show_numeric', 'gender', palette='deep', dodge=
grid.add_legend();
```



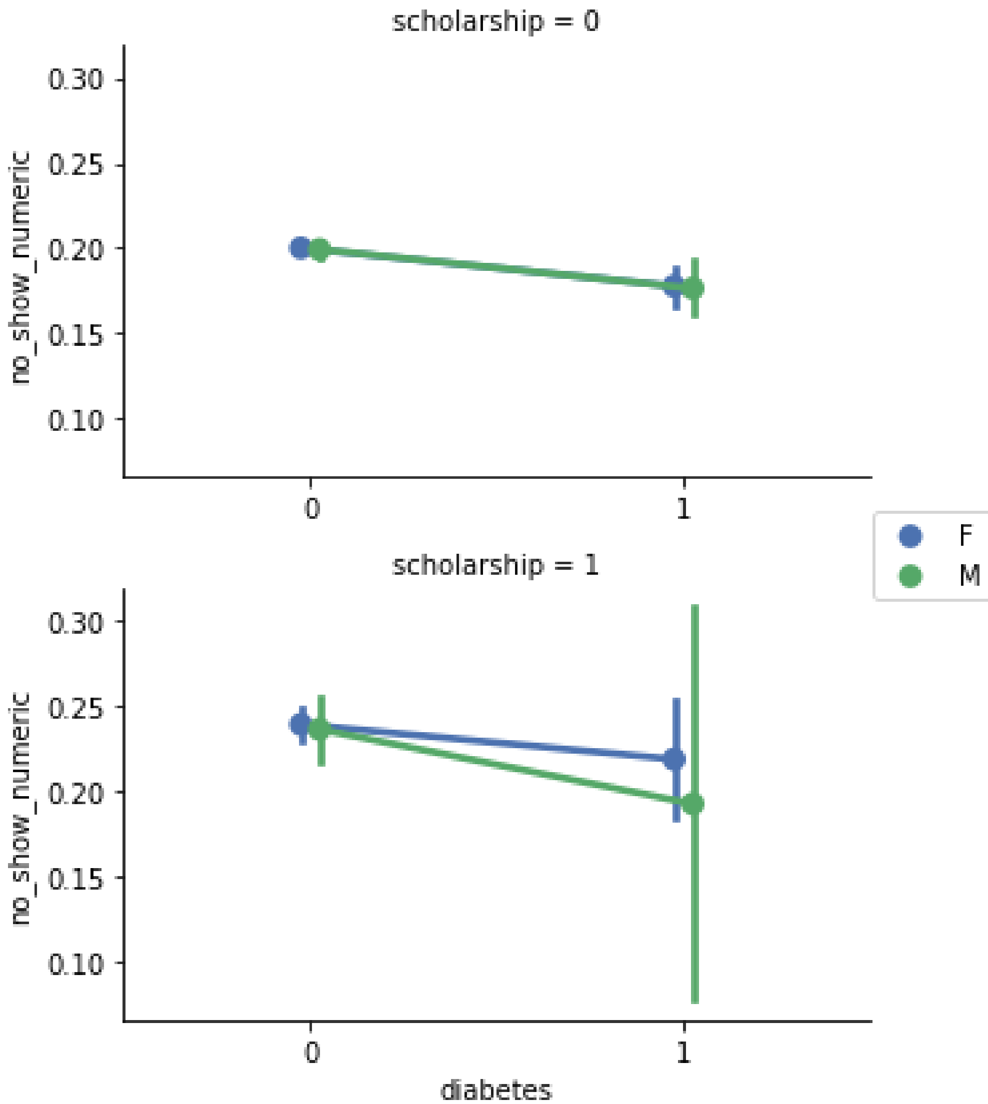
It is tempting (and counter intuitive at the same time) to say that patients with diabetes showed up less frequently if they got SMS messages. Unfortunately the standard error for diabetes is too high, lowering our confidence in the result showed in the diagram above.

```
In [50]: # Appointments related to patients with hypertension and receiving SMS:
         grid = sns.FacetGrid(df, row='sms_received', aspect=1.6)
         grid.map(sns.pointplot, 'hypertension', 'no_show_numeric', 'gender', palette='deep', do
         grid.add_legend();
```

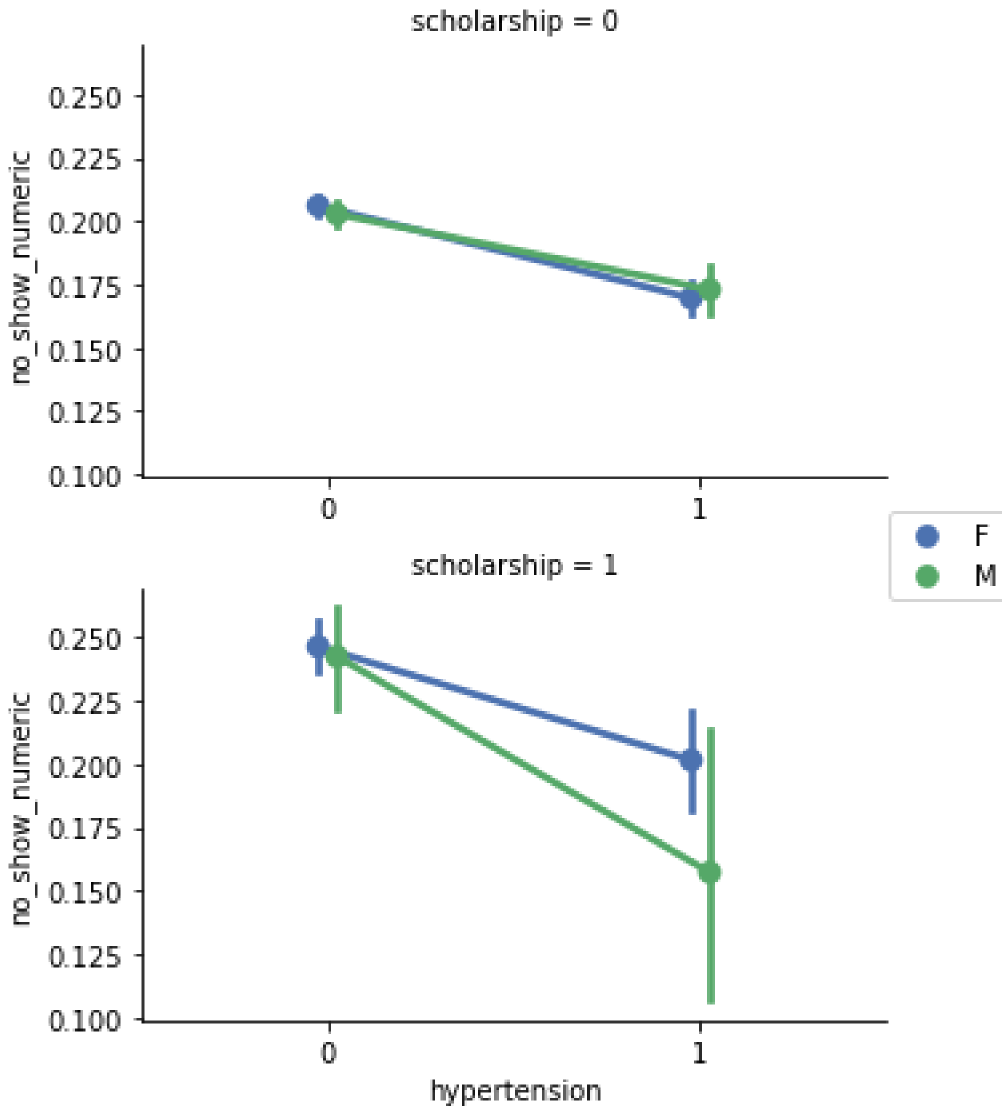


Among appointments done by patients with hypertension, it seems that sending SMS results in a greater ratio of no-shows (22-24% comparing to 16%). Another surprising insight tell us that men's no-show ratio is lower than women's. Normally, I would say otherwise. In both cases (when SMS is sent or not), no-show ratio of patients with diagnosed hypertension is lower than the ones without hypertension.

```
In [51]: # Appointments related to patients with diabetes and participation in scholarship:
         grid = sns.FacetGrid(df, row='scholarship', aspect=1.6)
         grid.map(sns.pointplot, 'diabetes', 'no_show_numeric', 'gender', palette='deep', dodge=
         grid.add_legend();
```




```
In [52]: # Appointments related to patients with hypertension and participation in scholarship:
         grid = sns.FacetGrid(df, row='scholarship', aspect=1.6)
         grid.map(sns.pointplot, 'hypertension', 'no_show_numeric', 'gender', palette='deep', do
         grid.add_legend();
```



Regarding scholarship, the result cannot be interpreted because of too high standard error span.

3 Conclusions

I have looked into the dataset and managed a few problems like unifying names, removing wrong data, adding new features based on existing data. I have also investigated most of independent variables in the dataset and made a few observations comparing them to each other as well as to the dependent one (no_show). As this was only an exploratory analysis, many potential correlations may remain uncovered. The data should be investigated further with more advanced statistical analysis to potentially reveal new insights and correlations.

The most important findings are:

Scheduling visits started on 2015-11-10 and ended on 2016-06-08. Visit appointments started on 2016-04-29 and ended on 2016-06-08.

The distribution of appointments among days of week (Monday-Friday) is almost equal with a little bit less visits on Thursday and Friday. There are 24 visits on Saturday and none on Sunday.

10 days on average patients awaited for an appointment. 50% of patients waited up to 4 days and 75% up to 15 days for an appointment. The longest awaiting time was 179 days. Almost 40k patients scheduled their visit for the same day. Out of all patients scheduling an appointment for the same day (in total 38561), 1792 of patients did not show up (5%).

There are many very young people in the dataset (most of them of age 0) but in general the patients age is distributed evenly and the number of patients goes drastricly down for patients older than 60 years.

The patients are 37 years on average. 25% of patients are below 18 and most of them are below 55.

Most of the patients are not alcoholics.

Most of the patients are not diabetes but more than alcoholics. There are for handicap categories with most of the people not being handicapted. Most patients do not have hypertension diagnosed.

On average, 20% of appointments were missed.

Out of 71831 appointments made by females, 14588 were missed with the ratio of 20%. Out of 38685 appointments made by males, 7723 were missed with the ratio of 20%.

There are patients with multiple appoinpments. The number appointments of top 10 patients range from 88 to 55. Taking into consideration, that the time range of visits appointed spans over 3 months, an appointment is most likely each examination or each specialist visit. So within one patient visit in a hospital, there could be multiple appointments scheduled. One of the no-show reasons could be the fact, that patients could be too tired to take part in all examinations during a particular visit, or the open hours were not sufficient to show up in all appointments. There could be also other reasons. The high number of appointments over so short period of time should be consulted with an SME to perform (or not) additional analysis in this area.

For all categorical variables the distributions of show / no-show for different categories look very similar. There is no clear indication of any of these variables having bigger then others impact on show / no-show characteristics. The charts confirm about 20% no-show rate for most categories.

```
In [ ]:
```