Investigate_a_Dataset

October 29, 2022

1 Project: Investigate a Dataset - [Scheduled Hospital Visitation in Brazil]

2 Table of Contents

3

3.1 Introduction

3.1.1 Dataset Description

This dataset collects information from 100k medical appointments in Brazil and is focused on the question of whether or not patients show up for their appointment. A number of characteristics about the patient are included in each row. Dataset column attributes and decriptions include: ### Attributes and Description 0. PatientId: Unique way to identify a patient. 1. AppointmentID: Identification of each appointment. 2. Gender: Sex of patient either Male or Female. 3. ScheduledDay: Day a patient picks to visit the hospital. 4. AppointmentDay:Day a patient is to visit the hospital. 5. Age: How old is the patient. 6. Neighbourhood: Location of the hospital or actual place for appointment. 7. Scholarship: True or False . Indicates whether or not the patient is enrolled in Brasilian welfare program Bolsa Família. 8. Hipertension: True or False. 9. Diabetes: True or False. 10. Alcoholism: True or False. 11. Handcap: True or False. 12. SMS_received: whether patient received SMS or not. Received or Not_received. 13. No-show: Yes or No. No implying the patient showed for their apppointment while Yes imply they did not.

Male = M, **Female** = F, **True** = 1, **False** = 0, **Received** = 1, **Not_received** = 0

3.1.2 Question(s) for Analysis

1. Percentage of patients that shows up for appointments ?
2. Which patient gender will show up more for appointments ?
3. Does SMS_reminders helps patient show up for their appointments ?

```
In [1]: import pandas as pd
```

```
In [2]: from subprocess import call
        call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[2]: 0
```

```
In [3]: cd Database_No_show_appointments
```

```
/home/workspace/Database_No_show_appointments
```

```
In [4]: # Import necessary packages
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        % matplotlib inline
```

```
In [5]: # Upgrade pandas to use dataframe.explode() function.
        !pip install --upgrade pandas==0.25.0
```

```
Requirement already up-to-date: pandas==0.25.0 in /opt/conda/lib/python3.6/site-packages (0.25.0
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in /opt/conda/lib/python
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /opt/conda/lib/python3.6/site-p
Requirement already satisfied, skipping upgrade: numpy>=1.13.3 in /opt/conda/lib/python3.6/site-
Requirement already satisfied, skipping upgrade: six>=1.5 in /opt/conda/lib/python3.6/site-packa
```

Data Wrangling

> **Tip**: In this section of the report, you will load in the data, check for cleanliness, and then trim and clean your dataset for analysis. Make sure that you **document your data cleaning steps in mark-down cells precisely and justify your cleaning decisions.**

Load Data

```
In [6]: # Load noshowappointments dataset and print out 5 lines
        df=pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
        df.head(5)
```

```
Out[6]:       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

                AppointmentDay  Age      Neighbourhood  Scholarship  Hipertension  \
        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  Handcap  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 [7]: # View a concise summary of the dataset
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
PatientId         110527 non-null float64
AppointmentID     110527 non-null int64
Gender            110527 non-null object
ScheduledDay      110527 non-null object
AppointmentDay    110527 non-null object
Age               110527 non-null int64
Neighbourhood     110527 non-null object
Scholarship       110527 non-null int64
Hipertension      110527 non-null int64
Diabetes          110527 non-null int64
Alcoholism        110527 non-null int64
Handcap           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
```

Accessing Data

```
In [8]: #explore data
        df.tail(6)
```

```
Out[8]:            PatientId  AppointmentID Gender          ScheduledDay  \
        110521  3.635534e+13        5651072      F  2016-05-03T08:23:40Z
        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  Hipertension  \
110521  2016-06-07T00:00:00Z   53   MARIA ORTIZ            0             0
110522  2016-06-07T00:00:00Z   56   MARIA ORTIZ            0             0
110523  2016-06-07T00:00:00Z   51   MARIA ORTIZ            0             0
110524  2016-06-07T00:00:00Z   21   MARIA ORTIZ            0             0
110525  2016-06-07T00:00:00Z   38   MARIA ORTIZ            0             0
110526  2016-06-07T00:00:00Z   54   MARIA ORTIZ            0             0

        Diabetes  Alcoholism  Handcap  SMS_received No-show
110521         0           0        0             1      No
110522         0           0        0             1      No
110523         0           0        0             1      No
110524         0           0        0             1      No
110525         0           0        0             1      No
110526         0           0        0             1      No
```

```
In [9]: #check sample size and no of column
        df.shape
```

```
Out[9]: (110527, 14)
```

```
In [10]: #check column names
         df.columns
```

```
Out[10]: Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
                'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hipertension',
                'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'No-show'],
               dtype='object')
```

```
In [11]: #Check for duplicated data
         df.duplicated().sum().any()
```

```
Out[11]: False
```

```
In [12]: # check for null values
         df.isnull().sum().any()
```

```
Out[12]: False
```

```
In [13]: #check for the data types
         df.dtypes
```

```
Out[13]: 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 [14]:
```
# Check for the number of unique value in each column
df.nunique()
```

Out[14]:
```
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 [15]:
```
# summary statistics of the data
df.describe()
```

Out[15]:

	PatientId	AppointmentID	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	3.921784e+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	

	Hipertension	Diabetes	Alcoholism	Handcap	\
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
count  110527.000000
mean        0.321026
std         0.466873
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
```

Observations The dataset has a sample size of 110527 and 14 columns. The dataset has no duplicated data and null values. Appointment day and Schedule day have wrong datatype. Patient age is within the range -1(min) and 115(max) with a mean value of 37. Minimum age seems inappropriate as we don't get age of -1 in actual sense. Approximately 9.8Approximately 19.7Approximately 7.18Approximately 3Approximately 22Approximately 32

3.1.3 Data Cleaning

1. Dropping columns not necessary for the analysis. 2. Rename column names to more intuitive and appropriate names. 3. Convert all column names to lower case for consistency. 4. Change value names(quantitative variables) to more intuitive names(categorical variables). 5. Datetime formatting. 6. Delete rows with inappropriate values e.g Age with a minimum value of -1.

```
In [16]: # Drop columns not necessary for the analysis
         df.drop(['PatientId', 'AppointmentID', 'Handcap'], axis=1, inplace=True)
```

```
In [17]: #confirm drop changes
         df.columns
```

```
Out[17]: Index(['Gender', 'ScheduledDay', 'AppointmentDay', 'Age', 'Neighbourhood',
                'Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', 'SMS_received',
                'No-show'],
               dtype='object')
```

```
In [20]: #Renaming column names
         df = df.rename(columns={'Hipertension' : 'Hypertension','Scholarship': 'BolsaFamília','
```

```
In [21]: #confirm changes
         df.columns
```

```
Out[21]: Index(['Gender', 'ScheduledDay', 'AppointmentDay', 'Age', 'Neighbourhood',
                'BolsaFamília', 'Hypertension', 'Diabetes', 'Alcoholism', 'SMS',
                'Appointment'],
               dtype='object')
```

```
In [22]: # The lambda function will be employed to avoid repetition
         # convert all column names to lower case for consistency
         df.rename(columns = lambda x : x.lower(), inplace=True)
         df.head(0)
```

```
Out[22]: Empty DataFrame
         Columns: [gender, scheduledday, appointmentday, age, neighbourhood, bolsafamília, hyper
         Index: []

In [23]: #change quantitative values in columns to more intuitive names(categorical variables)
         df['hypertension'] = df['hypertension'].replace([0,1],['Not_HTN', 'HTN'])
         df['diabetes'] = df['diabetes'].replace([0,1],['Not_DM', 'DM'])
         df['alcoholism'] = df['alcoholism'].replace([0,1],['Not_Alcoholic', 'Alcoholic'])
         df['bolsafamília'] = df['bolsafamília'].replace([0,1],['Not_Enrolled', 'Enrolled'])
         df['sms'] = df['sms'].replace([0,1],['Not_Received', 'Received'])

In [24]: #confirm changes
         df.head()

Out[24]:   gender          scheduledday        appointmentday  age      neighbourhood  \
         0      F  2016-04-29T18:38:08Z  2016-04-29T00:00:00Z   62    JARDIM DA PENHA
         1      M  2016-04-29T16:08:27Z  2016-04-29T00:00:00Z   56    JARDIM DA PENHA
         2      F  2016-04-29T16:19:04Z  2016-04-29T00:00:00Z   62      MATA DA PRAIA
         3      F  2016-04-29T17:29:31Z  2016-04-29T00:00:00Z    8  PONTAL DE CAMBURI
         4      F  2016-04-29T16:07:23Z  2016-04-29T00:00:00Z   56    JARDIM DA PENHA

            bolsafamília hypertension diabetes     alcoholism           sms appointment
         0  Not_Enrolled          HTN   Not_DM  Not_Alcoholic  Not_Received          No
         1  Not_Enrolled      Not_HTN   Not_DM  Not_Alcoholic  Not_Received          No
         2  Not_Enrolled      Not_HTN   Not_DM  Not_Alcoholic  Not_Received          No
         3  Not_Enrolled      Not_HTN   Not_DM  Not_Alcoholic  Not_Received          No
         4  Not_Enrolled          HTN       DM  Not_Alcoholic  Not_Received          No

In [25]: # change the 'No' and 'Yes' to more intuitive names(categorical variables)
         df['appointment'] = df['appointment'].replace(['No', 'Yes'],['show','Not_show'])

In [26]: #confirm changes
         df['appointment'].unique()

Out[26]: array(['show', 'Not_show'], dtype=object)

In [27]: #Datetime formatting
         #Introducing lambda to remove time value as is not necessary for this analysis
         df[['scheduledday','appointmentday']] = df[['scheduledday','appointmentday']].astype('d

In [28]: #confirm changes
         df[['scheduledday','appointmentday']].dtypes

Out[28]: scheduledday      datetime64[ns]
         appointmentday    datetime64[ns]
         dtype: object

In [29]: #Delete rows with wrong values(age < 0)
         wrong_age =df.query('age =="-1"').index
```

```
In [30]: df.drop(wrong_age,axis=0, inplace=True)

In [31]: #confirm for roll with age < 0
         df.query('age =="-1"')

Out[31]: Empty DataFrame
         Columns: [gender, scheduledday, appointmentday, age, neighbourhood, bolsafamília, hyper
         Index: []

In [32]: #check min age to confirm if row with inappropiate value is removed
         df.age.min()

Out[32]: 0
```

Exploratory Data Analysis

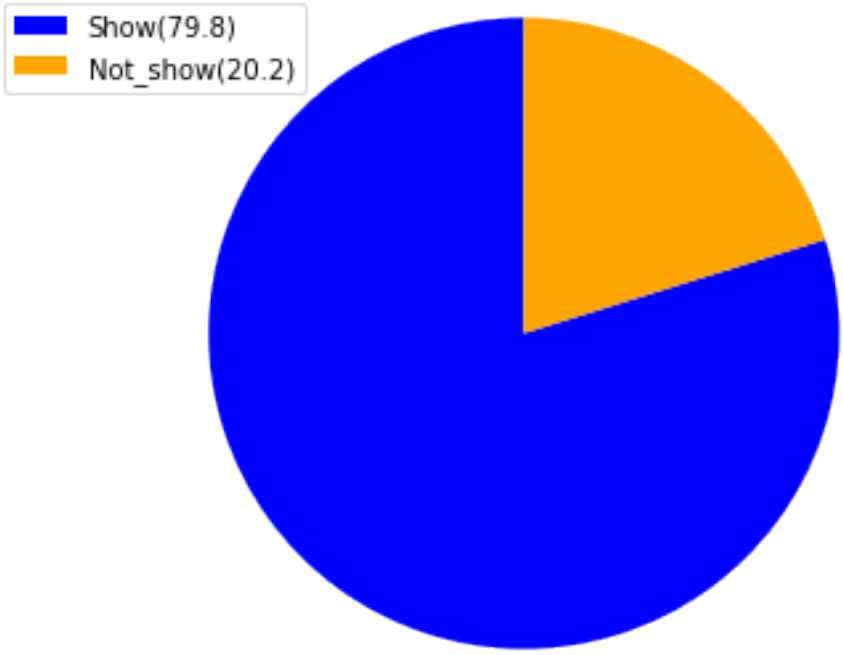
3.1.4 Research Question 1 (Percentage of patients that shows up for appointment)

```
In [33]: # percentage of patients that showed up for appointment
         percentage_count = df['appointment'].value_counts()/df['appointment'].count()*100
         percentage_count.round(1)

Out[33]: show        79.8
         Not_show    20.2
         Name: appointment, dtype: float64

In [34]: #chart labels
         labels = [r'Show(79.8)', r'Not_show(20.2)']
         sizes = [79.8, 20.2]
         colors =['blue', 'orange']
         patches, texts = plt.pie(sizes, colors=colors, startangle=90)
         plt.legend()
         plt.legend(patches, labels, loc="best")

         # Set aspect ratio to be equal so that pie is drawn as a circle.
         plt.axis('equal')
         plt.tight_layout()
         plt.show()
```



Approximately only 80% of the patients showed up for their scheduled appointment.

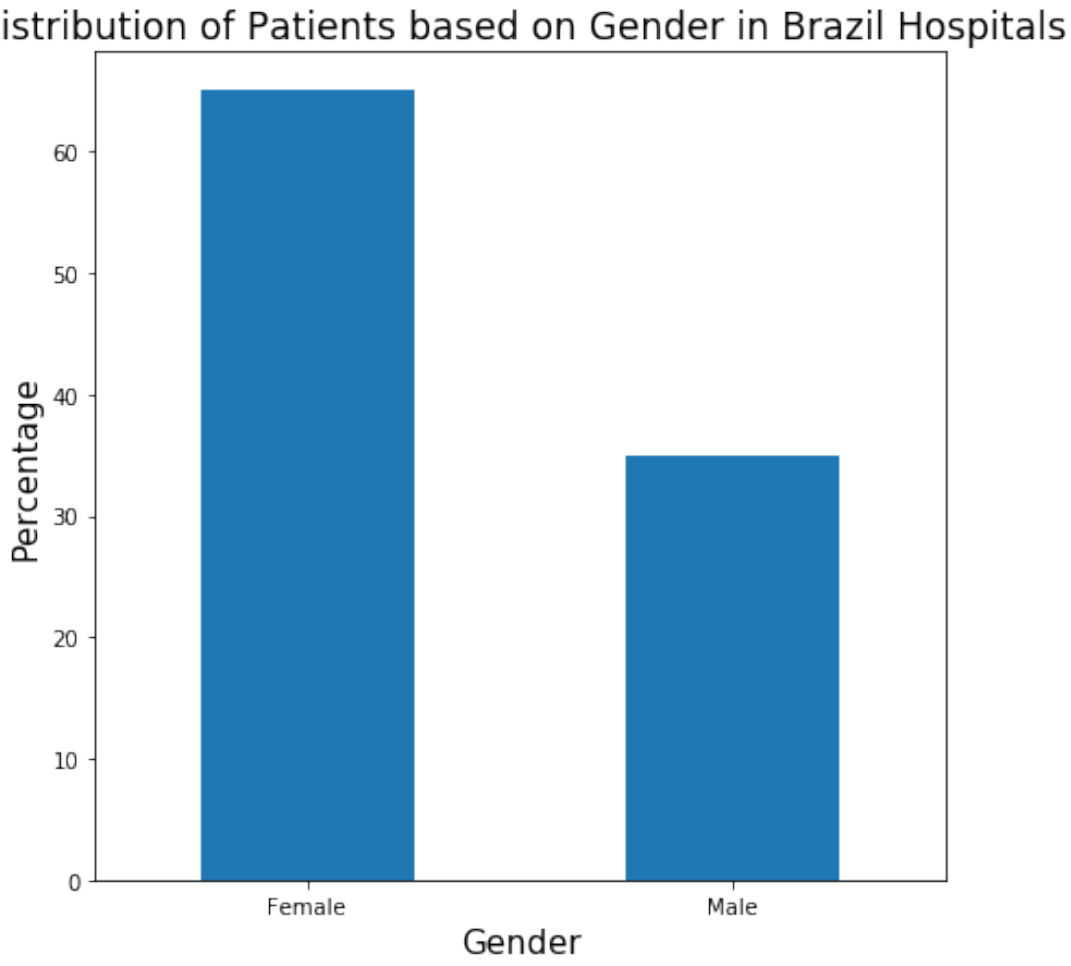
3.1.5 Research Question 2 (Which patients gender will show up more for scheduled appointment)

```
In [35]: # percentage of the patients by gender
         gender_percentage = df['gender'].value_counts()/df['gender'].count()*100
         gender_percentage.round(2)
```

```
Out[35]: F    65.0
         M    35.0
         Name: gender, dtype: float64
```

```
In [36]: # Explore gender distibution
         gender_percentage.plot(kind='bar', figsize=(7,7));

         # chart labels
         plt.xlabel('Gender', fontsize=15);
         plt.ylabel('Percentage', fontsize=15)
         plt.title('Distribution of Patients based on Gender in Brazil Hospitals', fontsize=17);
         location = [0,1]
         labels = ['Female', 'Male']
         plt.xticks(location, labels, rotation=0);
```



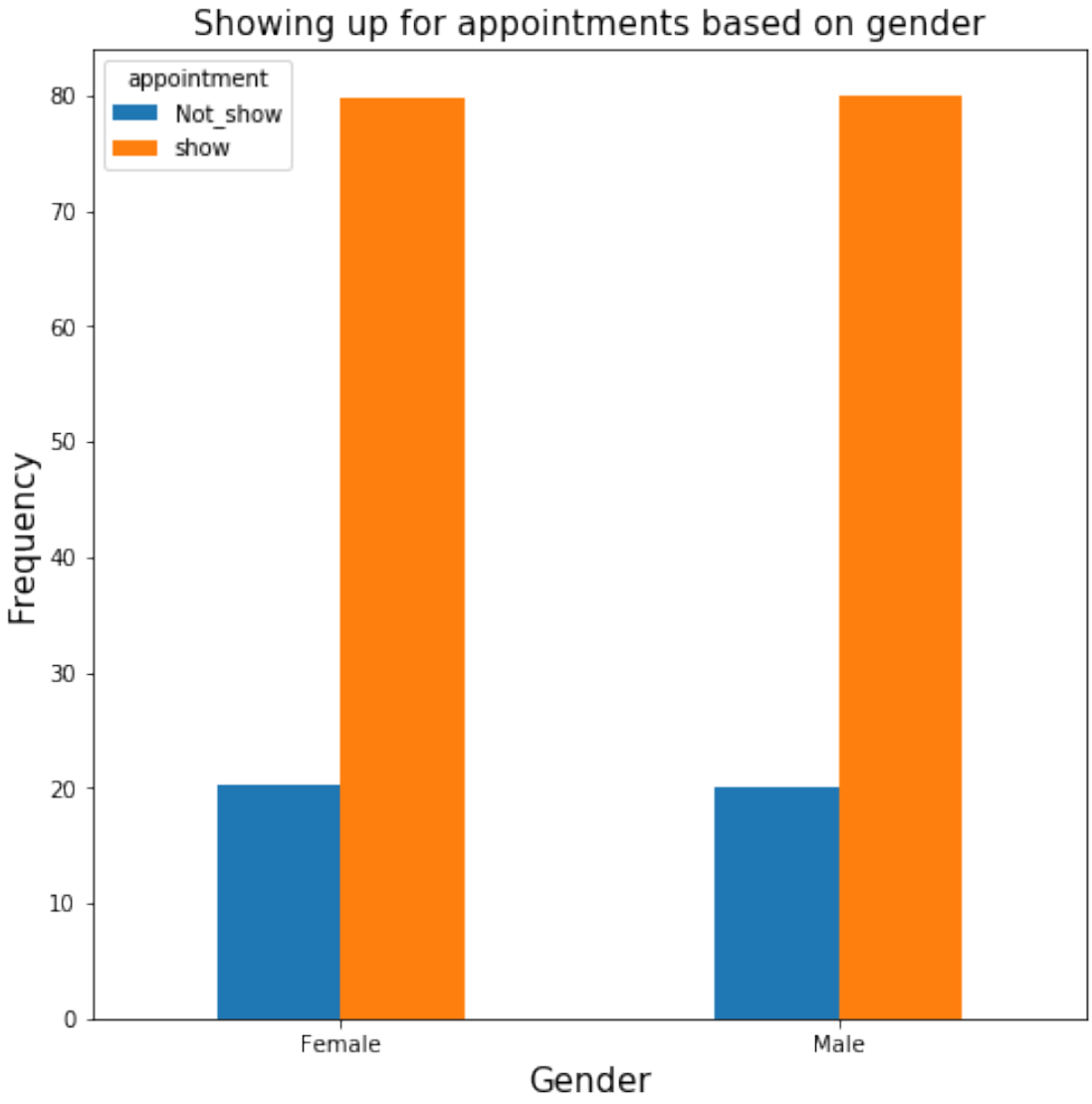

```
In [37]: # calculate the percentage of patients who showed up and not based on gender
         gender_percentage = df.groupby(['gender', 'appointment'])['age'].count()/df.groupby(['g
         gender_percentage.round(2)

Out[37]: gender  appointment
         F       Not_show       20.31
                 show           79.69
         M       Not_show       19.97
                 show           80.03
         Name: age, dtype: float64

In [38]: #Create a function to plot bar chart in other to acoid repetition
         def bar(data, xaxis, yaxis, titles, labels):
             data.plot(kind = 'bar', figsize =(8,8))
             plt.xlabel(xaxis, fontsize = 15)
             plt.ylabel(yaxis, fontsize = 15)
             plt.title(titles, fontsize = 15)
```

```
location = [0,1]
labels = labels
plt.xticks(location, labels, rotation = 0)
plt.show()
```

```
In [39]: # Double bar chart showing patient those who showed up and not based on gender
         # Using the bar function created to avoid repetition
         bar(gender_percentage.unstack(), 'Gender', 'Frequency', 'Showing up for appointments ba
```



3.1.6 Observation

From distribution of patients based on gender using percentage, it is observed that 65% of female show up for scheduled appointment while 35% of male show up for scheduled appointment.

From the comparison made between the male and female gender it can be inferred that the female gender value healthcare more than the male, but approximately same percentage of male and female show up for scheduled appointments.

3.1.7 Research Question 3 (Does SMS_reminders helps patient show up for their appointments)

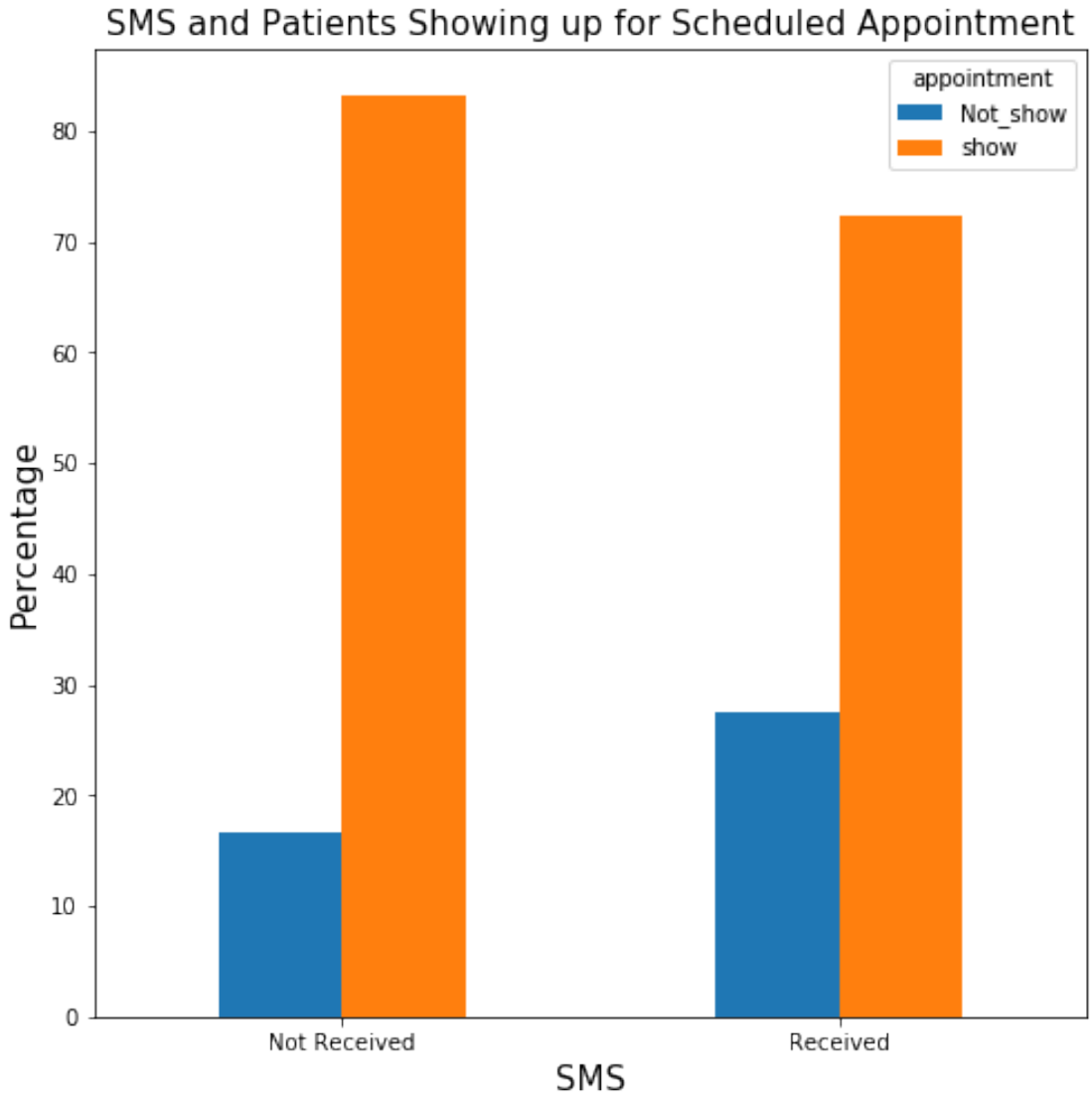
```
In [40]: # Percentage of patients that received sms
         percentage_sms = df['sms'].value_counts()/df['sms'].count()*100
         percentage_sms.round(2)

Out[40]: Not_Received    67.9
         Received        32.1
         Name: sms, dtype: float64

In [41]: # Percentage of all patients that received SMS and showed up
         percentage_sms_received = df.groupby(['sms', 'appointment'])['age'].count()/df.groupby(
         percentage_sms_received.unstack().round(1)

Out[41]: appointment   Not_show  show
         sms
         Not_Received      16.7  83.3
         Received          27.6  72.4

In [42]: # Double bar to show percentage of patients that received and did not received sms and
         labels = ['Not Received', 'Received']
         bar(percentage_sms_received.unstack(), 'SMS', 'Percentage', 'SMS and Patients Showing u
```



Observation 1. Only about 32% of patients received SMS. 83.3% of patients who did not received sms showed up for their appointment regardless as compared to 72.4% who did not receive sms at all, with reasons not specify.

4

4.1 Conclusions

Summary 1. Approximately 802. Majority of the patients are female, yet based on percentage distribution, approximately same percenatage of male and female gender show up for their medical appointments. 3. Receiving sms did not really contribute tp patients showing up or not for their medical appointments.

Limitations 1. Indequate information on why only a small percentage of the population received sms.

Recommendation A random sampling method should be employed for adequate distribution of the sample size, as from this dataset there was a higher number of females than male. Comparison would be made adequately using proportion or percentage

```
In [43]: df.to_csv('df_final.csv')
```