Investigate_a_Dataset

September 28, 2022

1 Project: Medical Appointment No Shows Analysis

1.1 Table of Contents

Introduction
- Data Wrangling
- Exploratory Data Analysis
- Conclusions
- ## Introduction

1.1.1 Dataset Description

In this project we are facing a situation : A person makes a doctor appointment, receives all the instructions and not show. Who to blame ? The aim of this dataset who collects information from 100k medical appointments in Brazil is to know whether or not patients will show up for their appointment by highlighting the elements that will help us the most in this prediction.

This analysis aim to determine What factors are important for us to predict if a patient will show up for their scheduled appointment.

1.1.2 Questions for Analysis

- What factors are important for us to know in order to predict if a patient will show up for their scheduled appointment or not ?
- Is there any relationship between age and patients who showed-up or not ?;
- Is the Gender of the patient an important factor to predict if a patient will show up or not ?
- Does the fact of receiving an sms an important indicator to predict if a patient will show up ?

In [26]:
```
#Let's import all the necessary libaries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime

% matplotlib inline
```

Data Wrangling

```
In [27]: #We load our data

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

Out[27]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap	SMS_received	No-show
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	0	1	0	0	0	0	No
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	0	0	0	0	0	No
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA	0	0	0	0	0	0	No
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI	0	0	0	0	0	0	No
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	1	1	0	0	0	No

Let's check some description of our data

```
In [28]: #Let's see row and columns
         df.shape
```

Out[28]: (110527, 14)

```
In [29]: 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
```

Its seem like there is no missing values, but some type like the `ScheduledDay` and the `AppointmentDay` are not correct, so let's change to the corrects types

1.1.3 Data Cleaning

First let's remove some row that will not be usefull in our Dataframe

```
In [30]: # We will drop PatientId, AppointmentID, Neighbourhood

         df = df.drop(['PatientId','AppointmentID', 'Neighbourhood'], axis = 1)

In [31]: #check if these columns where well droped

         df.shape

Out[31]: (110527, 11)

In [32]: # See how dateframe looks like by now starting by the bottom
         df.tail()
```

Out[32]:

	Gender	ScheduledDay	AppointmentDay	Age	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap	SMS_received	No-show
110522	F	2016-05-03T09:15:35Z	2016-06-07T00:00:00Z	56	0	0	0	0	0	1	No
110523	F	2016-05-03T07:27:33Z	2016-06-07T00:00:00Z	51	0	0	0	0	0	1	No
110524	F	2016-04-27T16:03:52Z	2016-06-07T00:00:00Z	21	0	0	0	0	0	1	No
110525	F	2016-04-27T15:09:23Z	2016-06-07T00:00:00Z	38	0	0	0	0	0	1	No
110526	F	2016-04-27T13:30:56Z	2016-06-07T00:00:00Z	54	0	0	0	0	0	1	No

Now let's rename some columns title as we have some mistake

```
In [33]: #Lets rename the columns No-show to No_show
         df.rename(columns = {'No-show' : 'No_show', 'Hipertension': 'Hypertension',
                              'Handcap': 'Handicap',},inplace = True)
```

```
In [34]: #Lets check the new names
         df.columns
```

```
Out[34]: Index(['Gender', 'ScheduledDay', 'AppointmentDay', 'Age', 'Scholarship',
                'Hypertension', 'Diabetes', 'Alcoholism', 'Handicap', 'SMS_received',
                'No_show'],
               dtype='object')
```

```
In [35]: #Here we are converting ScheduledDay and AppointmentDay in a Datetime format
         df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
         df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])
```

```
In [36]: #Let's check the new types
         df.dtypes
```

```
Out[36]: Gender                     object
         ScheduledDay      datetime64[ns]
         AppointmentDay    datetime64[ns]
         Age                        int64
         Scholarship                int64
         Hypertension               int64
         Diabetes                   int64
         Alcoholism                 int64
         Handicap                   int64
         SMS_received               int64
         No_show                   object
         dtype: object
```

```
In [37]: df.describe()
```

```
Out[37]:                 Age    Scholarship   Hypertension       Diabetes  \
         count  110527.000000  110527.000000  110527.000000  110527.000000
         mean       37.088874       0.098266       0.197246       0.071865
         std        23.110205       0.297675       0.397921       0.258265
         min        -1.000000       0.000000       0.000000       0.000000
         25%        18.000000       0.000000       0.000000       0.000000
         50%        37.000000       0.000000       0.000000       0.000000
         75%        55.000000       0.000000       0.000000       0.000000
         max       115.000000       1.000000       1.000000       1.000000

                   Alcoholism       Handicap   SMS_received
         count  110527.000000  110527.000000  110527.000000
         mean        0.030400       0.022248       0.321026
         std         0.171686       0.161543       0.466873
         min         0.000000       0.000000       0.000000
         25%         0.000000       0.000000       0.000000
         50%         0.000000       0.000000       0.000000
         75%         0.000000       0.000000       1.000000
         max         1.000000       4.000000       1.000000
```

As we are seing we have some negatives Age that we should remove.

```
In [38]: #We select all the age below 0 and we drop them
         low_age = df.query("Age < 0")
         low_age
```

```
Out[38]:       Gender        ScheduledDay AppointmentDay  Age  Scholarship  \
         99832      F 2016-06-06 08:58:13     2016-06-06   -1            0

                Hypertension  Diabetes  Alcoholism  Handicap  SMS_received No_show
         99832              0         0           0         0             0      No
```

```
In [39]: #Here we will drop this line
         df.drop(99832, inplace=True)
```

For the rest of our analysis we would replace the value of the `No_show` column with 0 or 1 as in the other columns

```
In [40]: df['No_show'].replace({"No" : 1, "Yes" : 0},inplace = True)
```

```
In [41]: #Let's check for the new columns value

         df['No_show'].head()
```

```
Out[41]: 0    1
         1    1
         2    1
         3    1
         4    1
         Name: No_show, dtype: int64
```

Exploratory Data Analysis

```
In [42]: #Let's have some descriptiv statistics
         df.describe()
```

```
Out[42]:                 Age    Scholarship   Hypertension       Diabetes  \
         count  110526.000000  110526.000000  110526.000000  110526.000000
         mean       37.089219       0.098266       0.197248       0.071865
         std        23.110026       0.297676       0.397923       0.258266
         min         0.000000       0.000000       0.000000       0.000000
         25%        18.000000       0.000000       0.000000       0.000000
         50%        37.000000       0.000000       0.000000       0.000000
         75%        55.000000       0.000000       0.000000       0.000000
         max       115.000000       1.000000       1.000000       1.000000

                   Alcoholism       Handicap   SMS_received        No_show
         count  110526.000000  110526.000000  110526.000000  110526.000000
         mean        0.030400       0.022248       0.321029       0.798066
```

```
std         0.171686       0.161543       0.466874       0.401445
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       1.000000
50%         0.000000       0.000000       0.000000       1.000000
75%         0.000000       0.000000       1.000000       1.000000
max         1.000000       4.000000       1.000000       1.000000
```

Let's calculate the mean of some of the numerical sample of people have an handicap, are sick and have a sholarship

```
In [43]: df_mean = ['Scholarship','Hypertension','Alcoholism','Diabetes','SMS_received']
         for x in df_mean :
             print(df.groupby(x)['No_show'].mean())
```

```
Scholarship
0    0.801926
1    0.762637
Name: No_show, dtype: float64
Hypertension
0    0.790961
1    0.826980
Name: No_show, dtype: float64
Alcoholism
0    0.798052
1    0.798512
Name: No_show, dtype: float64
Diabetes
0    0.796370
1    0.819967
Name: No_show, dtype: float64
SMS_received
0    0.832965
1    0.724255
Name: No_show, dtype: float64
```

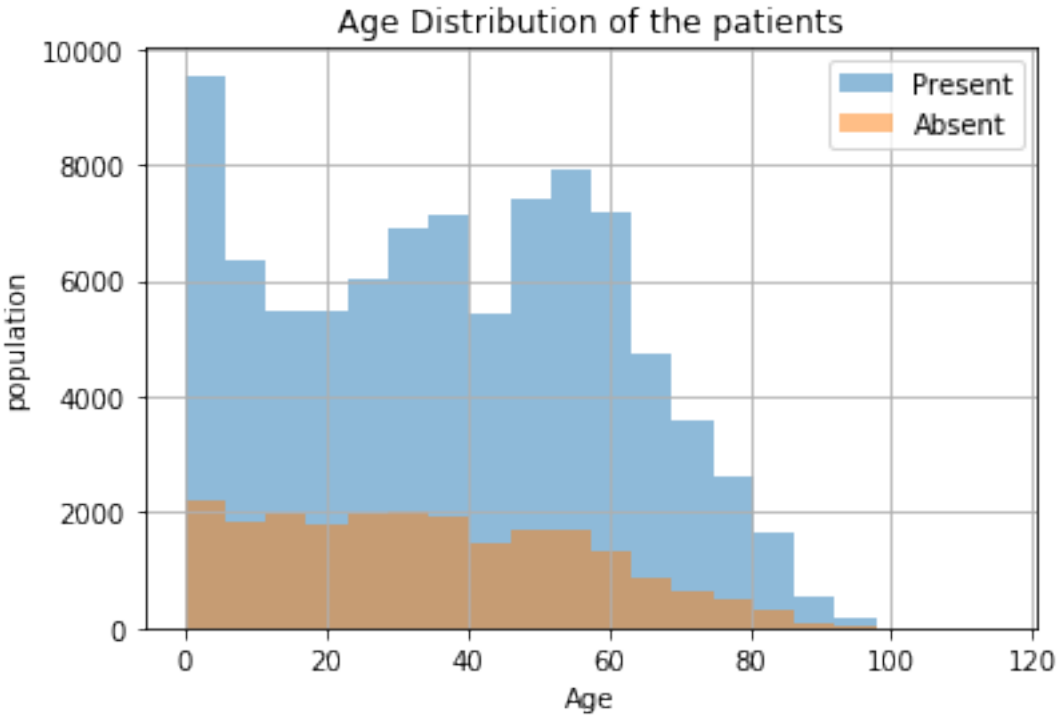
Let's create two mask, the first one for the people who attemp to their appointment and the second one for those who didnt attemp

```
In [44]: #Lets create some mask one for people who where present and
         #another one for those wo where absent
         #present = df[df['No_show'] == 1]
         #absent = df[df['No_show'] == 0]

         present = df.No_show == 1
         absent = df.No_show == 0
```

1.2 What are the important factor for our prediction ?

1.3 Let's look at the Age

```
In [45]: #Let's display an histogram showing relation with the age and the people who where pres

         df.Age[present].hist(alpha=0.5, bins=20, label='Present')
         df.Age[absent].hist(alpha=0.5, bins=20, label='Absent')
         plt.title('Age Distribution of the patients')
         plt.xlabel('Age');
         plt.ylabel('population');
         plt.legend();
```



As we can see there is age reduction in patients who did not Show up compared to those who did. Age between 0 and 10 are more likely to attemp to their schedule.

```
In [46]: #description of the people who where present
         df[present].Age.describe()

Out[46]: count    88207.000000
         mean        37.790504
         std         23.338645
         min          0.000000
         25%         18.000000
         50%         38.000000
         75%         56.000000
```

```
max         115.000000
Name: Age, dtype: float64
```

```
In [47]: #description of the people who where not present
         df[absent].Age.describe()
```

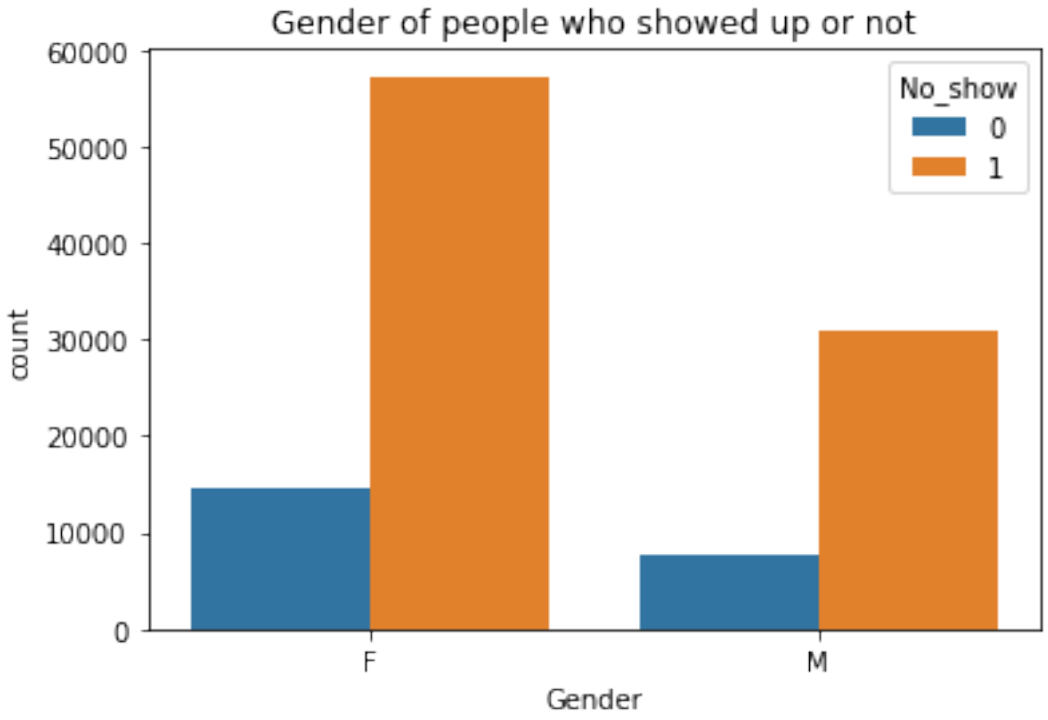
```
Out[47]: count    22319.000000
         mean        34.317667
         std         21.965941
         min          0.000000
         25%         16.000000
         50%         33.000000
         75%         51.000000
         max        115.000000
         Name: Age, dtype: float64
```

Here we can see and conclude that children from 0 to 10 are more likely the attemp to their schedule

1.4 Let's look at the Gender

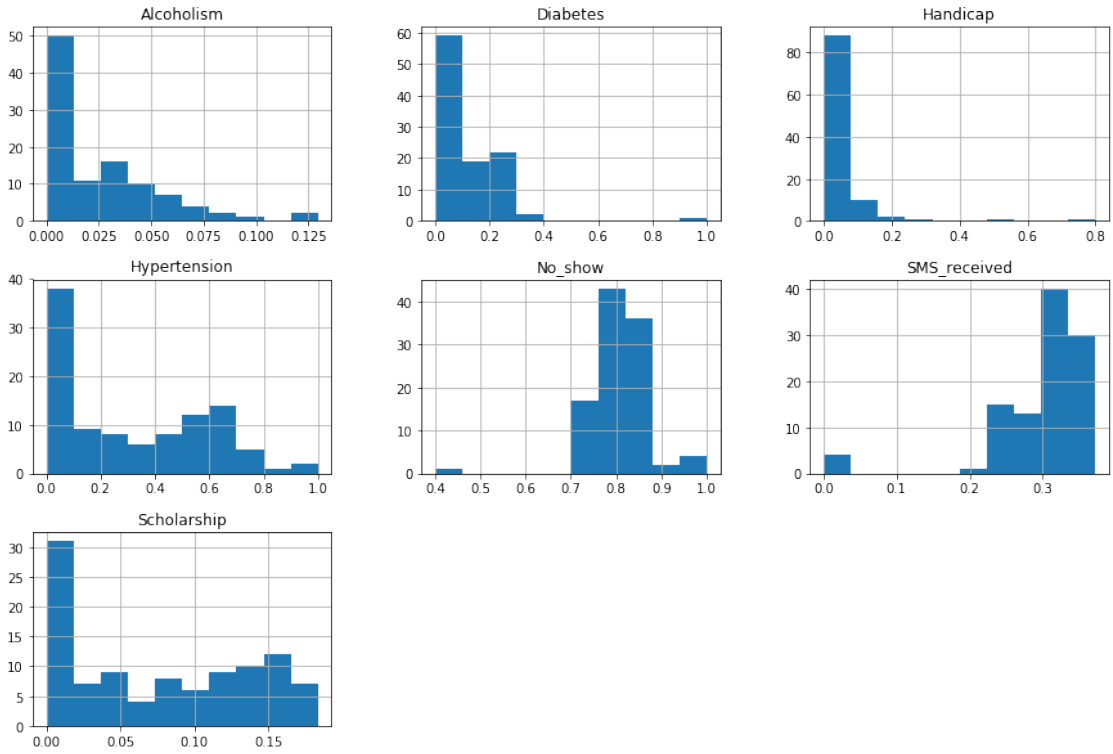
```
In [48]: #Lets make a plot function
         def Show_plot(col,title, hue_col = 'No_show', data = df):
             sns.countplot(x = col, data = data, hue = hue_col)
             plt.title(title);
```

```
In [49]: Show_plot(col = 'Gender',title = 'Gender of people who showed up or not')
```



As we can see on this graph, women are more likely to attemp their appointment

```
In [50]: #Let's select all the men who where present a their appointment
         df.loc[(df['Gender'] == 'M') & (df['No_show'] == 1)].count()

Out[50]: Gender            30962
         ScheduledDay      30962
         AppointmentDay    30962
         Age               30962
         Scholarship       30962
         Hypertension      30962
         Diabetes          30962
         Alcoholism        30962
         Handicap          30962
         SMS_received      30962
         No_show           30962
         dtype: int64

In [51]: #Let's select all the women who where present a their appointment
         df.loc[(df['Gender'] == 'F') & (df['No_show'] == 1)].count()

Out[51]: Gender            57245
         ScheduledDay      57245
         AppointmentDay    57245
         Age               57245
         Scholarship       57245
         Hypertension      57245
         Diabetes          57245
         Alcoholism        57245
         Handicap          57245
         SMS_received      57245
         No_show           57245
         dtype: int64

In [52]: #Let's check the exact repartition of men and dwomen who where present or not
         df.groupby('Gender')['No_show'].value_counts()

Out[52]: Gender  No_show
         F       1          57245
                 0          14594
         M       1          30962
                 0           7725
         Name: No_show, dtype: int64

In [53]: df.groupby('Age').mean().hist(figsize=(15,10));
```

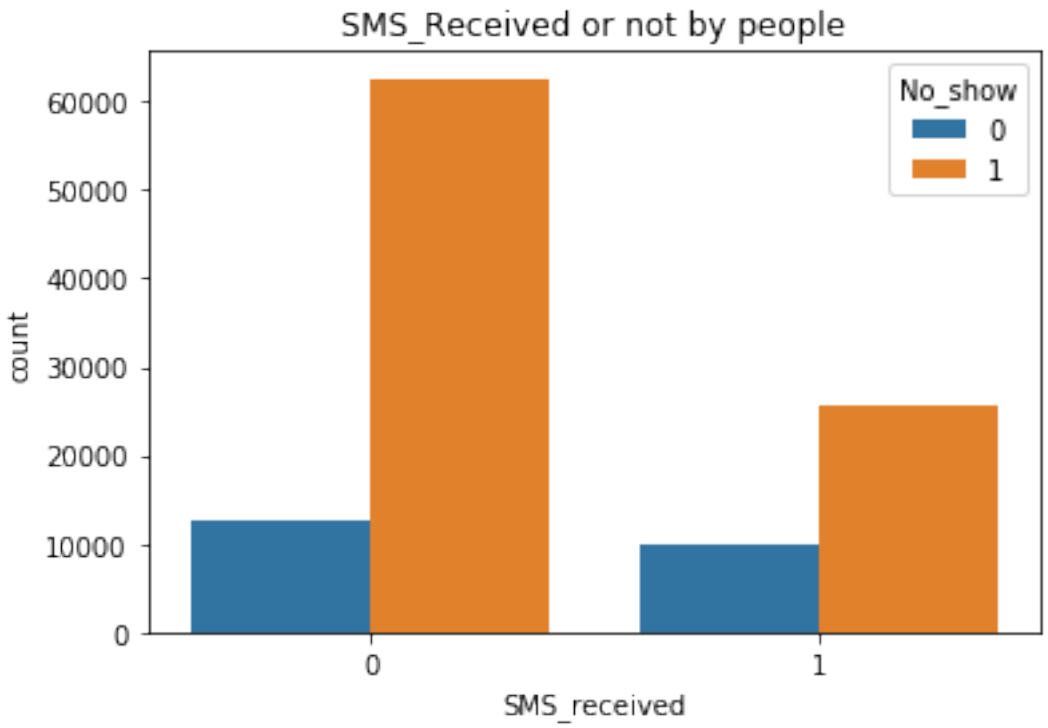


1.5 Lets look at the SMS received

```
In [34]: df['SMS_received'].value_counts()
```

```
Out[34]: 0    75044
         1    35482
         Name: SMS_received, dtype: int64
```

```
In [46]: Show_plot(col = 'SMS_received',title = 'SMS_Received or not by people')
```



Conclusions

From our analysis, we have discover that gender, age, and sms are important to predict if a patient will show up or not. we have discover that women are more likely to show-up to their appointment than male, although the data is imbalanced as we have more Females than Male.

The Age variation show us that children in the age between 0 to 10 are more likely to show up to their appointment when Certain others age groups appear to be more likely to miss their appointments. , this can be because they are babies with important medical attentions.

1.6 Limitations

- in terms of whether or not to receive a text message, a small uncertainty still remains
- we also had a negative value of age which was dropped not knowing why we had this kind of value at this place.

Having more domain knowledge about the data like the total of the population in this region, might have helped us to understand better.

```
In [53]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[53]: 0
```