Investigate_a_Dataset

October 21, 2021

1 Project: Investigate a Dataset - Noshow appointments

1.1 Table of Contents

Introduction
Data Wrangling
Exploratory Data Analysis
Conclusions
Introduction

1.1.1 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

'ScheduledDay' tells us on what day the patient set up their appointment.

'Neighborhood' indicates the location of the hospital.

'Scholarship' indicates whether or not the patient is enrolled in Brasilian welfare program Bolsa Família.

'No_show' it says 'No' if the patient showed up to their appointment, and 'Yes' if they did not show up.

1.1.2 Question(s) for Analysis

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

```
In [2]: # Use this cell to set up import statements for all of the packages that you
        #   plan to use.
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns


        # Remember to include a 'magic word' so that your visualizations are plotted
        #   inline with the notebook. See this page for more:
```

```
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html
% matplotlib inline
```

```
In [12]: # 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: pytz>=2017.2 in /opt/conda/lib/python3.6/site-p
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in /opt/conda/lib/python
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

1.1.3 General Properties

```
In [3]: # Load your data and print out a few lines. Perform operations to inspect data
        #   types and look for instances of missing or possibly errant data.
        df = pd.read_csv('noshowappointments.csv')
        df.head()
```

Out[3]:

	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

```
In [4]: #find the number of rows and columns
        df.shape
```

```
Out[4]: (110527, 14)
```

```
In [5]: #check the number of unique
        df['PatientId'].nunique()
```

```
Out[5]: 62299

In [6]: #check the number of duplicated PatientId
        df['PatientId'].duplicated().sum()

Out[6]: 48228

In [7]: #check the number of duplicated PatientId and no-show
        df.duplicated(['PatientId','No-show']).sum()

Out[7]: 38710
```

I will drop them

```
In [8]: #inspect data types
        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


In [9]: df.describe()

Out[9]:           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
```

The mean ages is 37 years
Maximum age is 115 years
There is a negative age and this does not make sense

1.1.4 Data Cleaning

```
In [10]: df.rename(columns = lambda x : x.replace("-","_"),inplace=True)
```

Rewrite - to _ so you don't make a mistake while writing the code for parsing

```
In [11]: # drop duplicated PatientId with duplicated show status
         df.drop_duplicates(['PatientId','No_show'], inplace=True)
```

Drop some duplicates to reduce the data and leave only the useful data

```
In [12]: #dropping the columns that are not useful for this investigation
         df.drop(['PatientId','AppointmentID','ScheduledDay','AppointmentDay'], axis = 1, inplac
         df.head()
```

```
Out[12]:   Gender  Age      Neighbourhood  Scholarship  Hipertension  Diabetes  \
         0      F   62    JARDIM DA PENHA            0             1         0
         1      M   56    JARDIM DA PENHA            0             0         0
         2      F   62      MATA DA PRAIA            0             0         0
         3      F    8  PONTAL DE CAMBURI            0             0         0
         4      F   56    JARDIM DA PENHA            0             1         1

            Alcoholism  Handcap  SMS_received No_show
         0           0        0             0      No
```

```
1           0        0             0      No
2           0        0             0      No
3           0        0             0      No
4           0        0             0      No
```

Drop some data that we don't need in the analysis

```
In [13]: # convert no_show values to 0=No and 1=Yes
         df['No_show'] = df['No_show'].replace({'No':0,'Yes':1})
         df.head()
```

```
Out[13]:   Gender  Age      Neighbourhood  Scholarship  Hipertension  Diabetes  \
         0      F   62    JARDIM DA PENHA            0             1         0
         1      M   56    JARDIM DA PENHA            0             0         0
         2      F   62      MATA DA PRAIA            0             0         0
         3      F    8  PONTAL DE CAMBURI            0             0         0
         4      F   56    JARDIM DA PENHA            0             1         1

            Alcoholism  Handcap  SMS_received  No_show
         0           0        0             0        0
         1           0        0             0        0
         2           0        0             0        0
         3           0        0             0        0
         4           0        0             0        0
```

Make no show equal to one and zero so that I can understand it clearly

```
In [14]: # drop negative ages
         neg_age = df[df.Age<0]
         df.drop(neg_age.index, inplace=True)
```

Remove the ages that are in the negative because they are illogical

```
In [15]: # rename columns
         df = df.rename(columns={'Hipertension':'hypertension','Handcap':'handicap'})
         df.head()
```

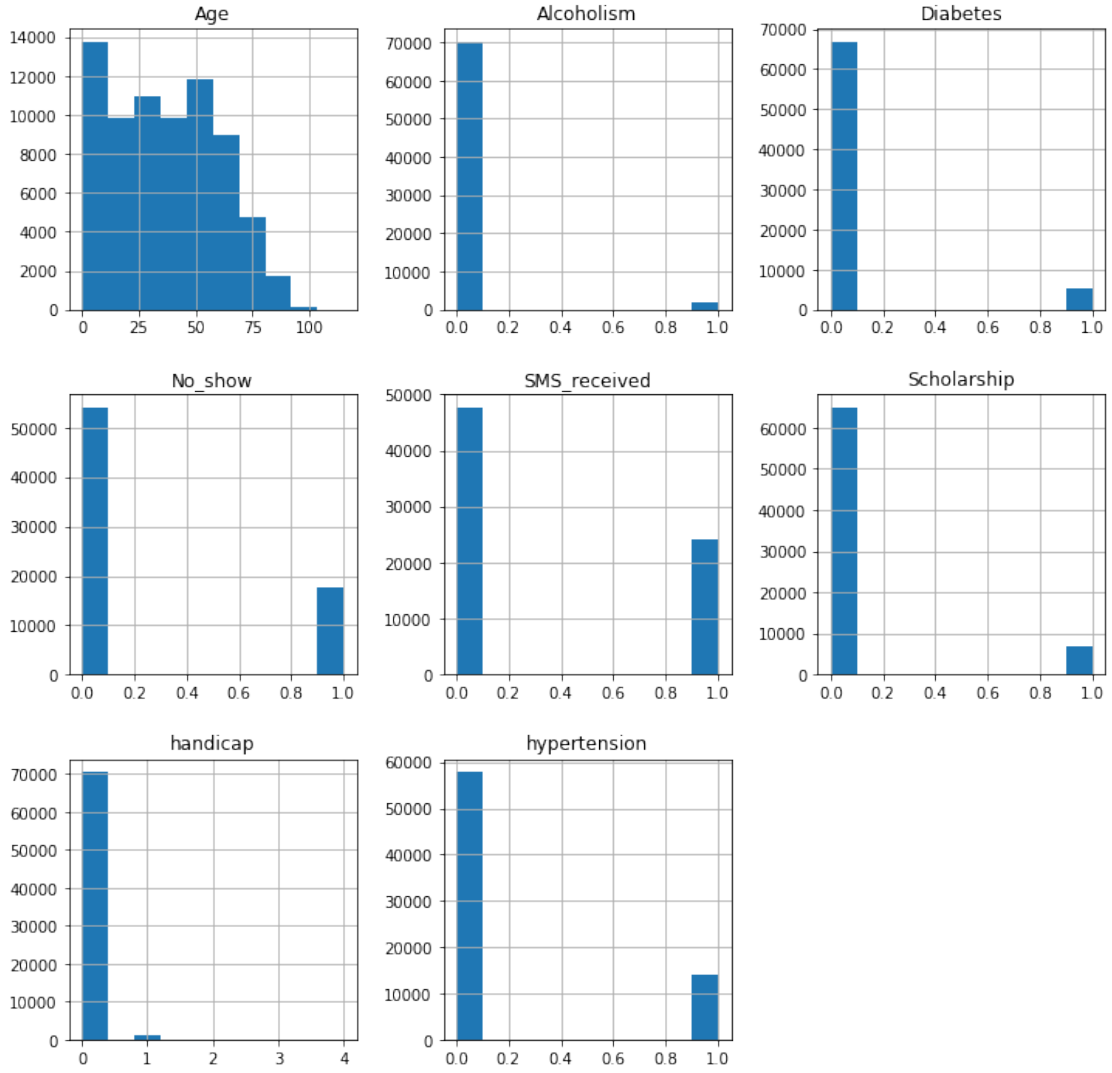
```
Out[15]:   Gender  Age      Neighbourhood  Scholarship  hypertension  Diabetes  \
         0      F   62    JARDIM DA PENHA            0             1         0
         1      M   56    JARDIM DA PENHA            0             0         0
         2      F   62      MATA DA PRAIA            0             0         0
         3      F    8  PONTAL DE CAMBURI            0             0         0
         4      F   56    JARDIM DA PENHA            0             1         1

            Alcoholism  handicap  SMS_received  No_show
         0           0         0             0        0
         1           0         0             0        0
         2           0         0             0        0
         3           0         0             0        0
         4           0         0             0        0
```

Rewriting some characteristics so that I don't make a mistake while writing the code to analyze and understand the data more clearly

Exploratory Data Analysis

1.1.5 Research Question 1: What factors are important for us to know in order to predict if a patient will show up for their scheduled appointment?

```
In [16]: # Use this, and more code cells, to explore your data. Don't forget to add
         #   Markdown cells to document your observations and findings.
         df.hist(figsize=(12,12));#histogram of dataset
```



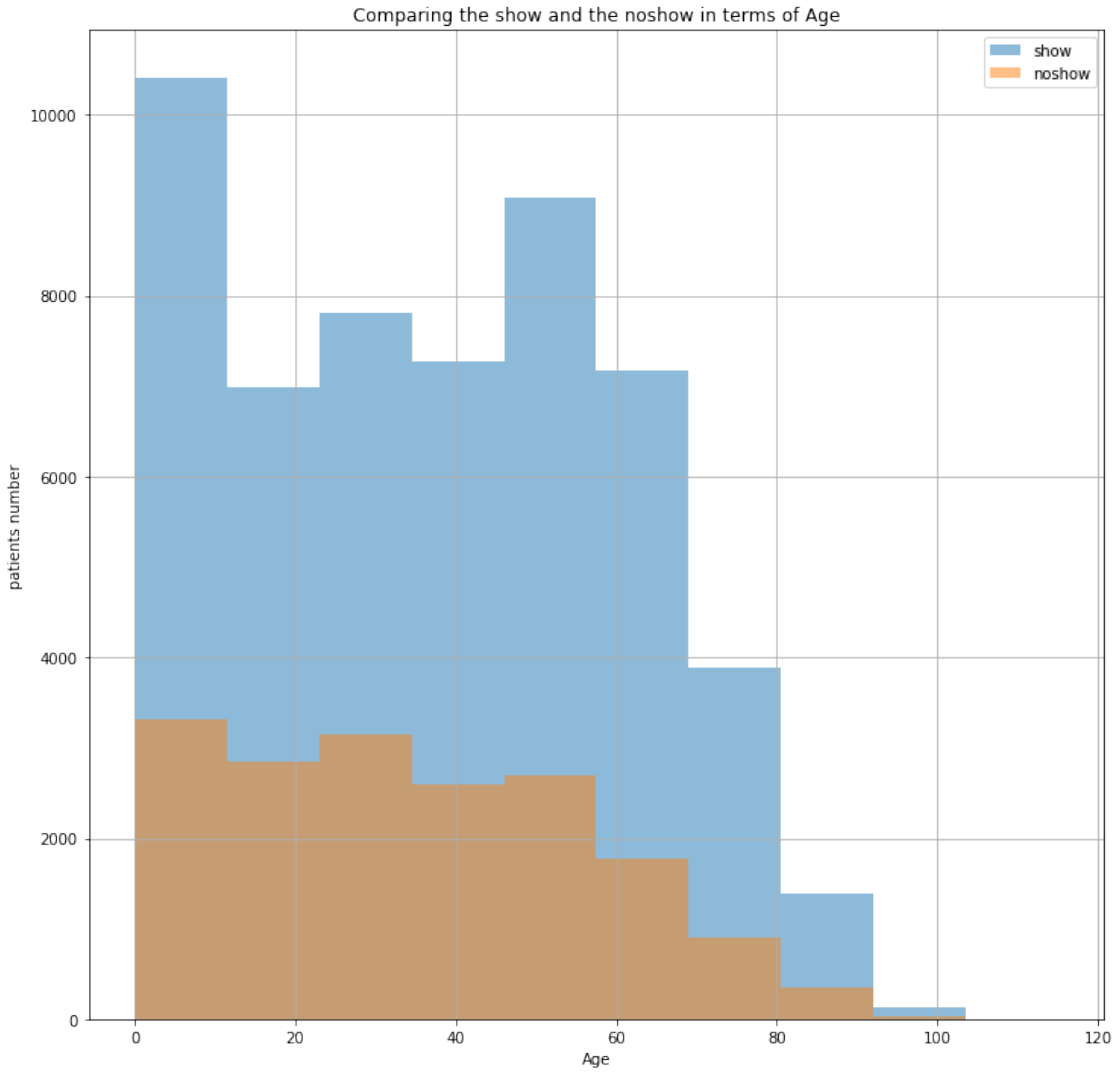
Most of the patients do not have diabetes and Handicap
Most patients are non-alcoholic
The number of people who received SMS is half of those who did not

```
In [17]: #assigning names
         show = df.No_show == 0
         noshow = df.No_show == 1

In [18]: df[show].count()

Out[18]: Gender           54153
         Age              54153
         Neighbourhood    54153
         Scholarship      54153
         hypertension     54153
         Diabetes         54153
         Alcoholism       54153
         handicap         54153
         SMS_received     54153
         No_show          54153
         dtype: int64

In [19]: df[noshow].count()

Out[19]: Gender           17663
         Age              17663
         Neighbourhood    17663
         Scholarship      17663
         hypertension     17663
         Diabetes         17663
         Alcoholism       17663
         handicap         17663
         SMS_received     17663
         No_show          17663
         dtype: int64
```

The number of the show is four times the number of the noshow

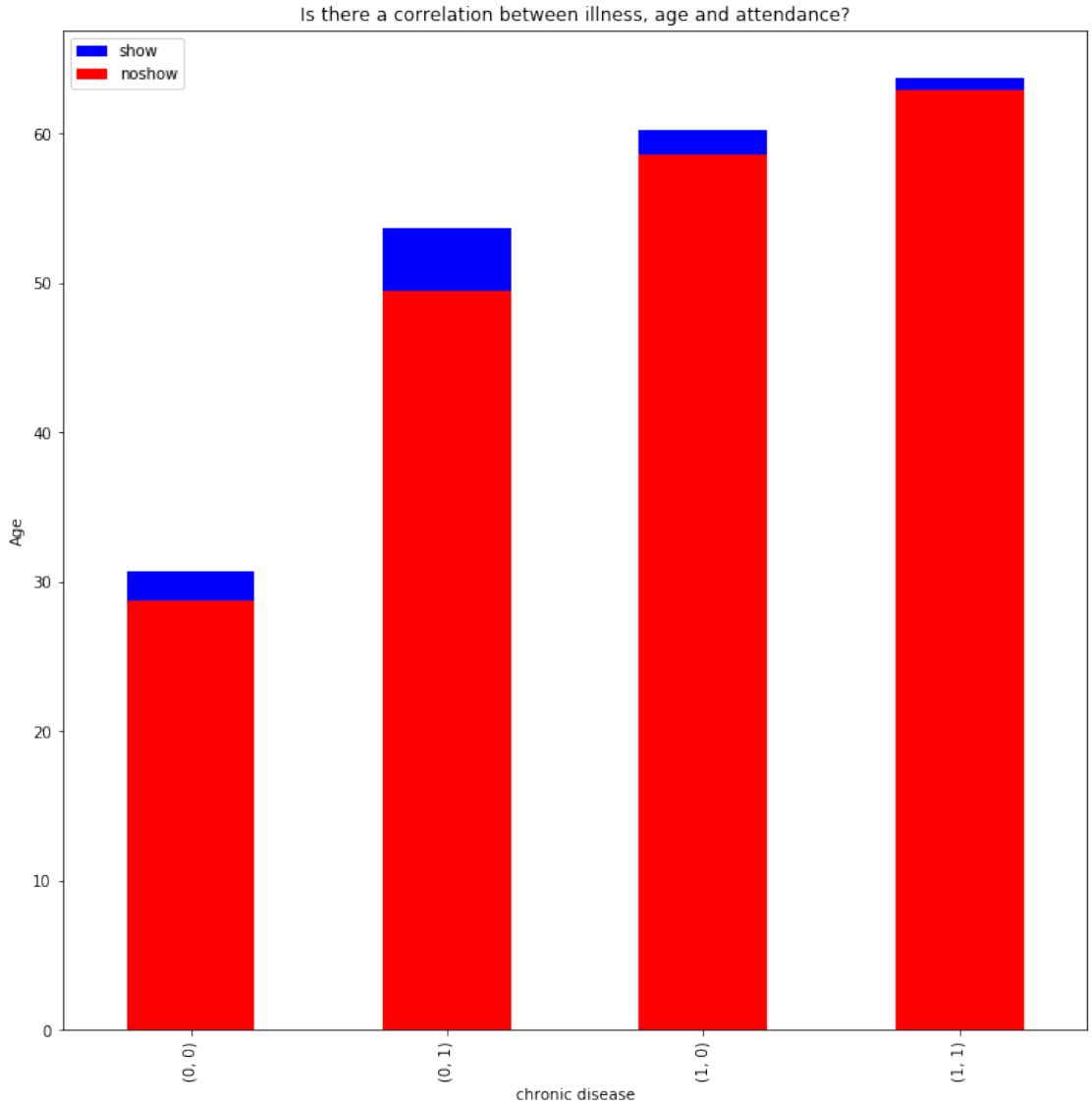
2 Analysing

```
In [20]: #Comparing the show and the noshow in terms of Age
         plt.figure(figsize=[12, 12])
         df.Age[show].hist(alpha=0.5,label='show')
         df.Age[noshow].hist(alpha=0.5,label='noshow')
         plt.legend()
         plt.title('Comparing the show and the noshow in terms of Age')
         plt.xlabel('Age')
         plt.ylabel('patients number');
```



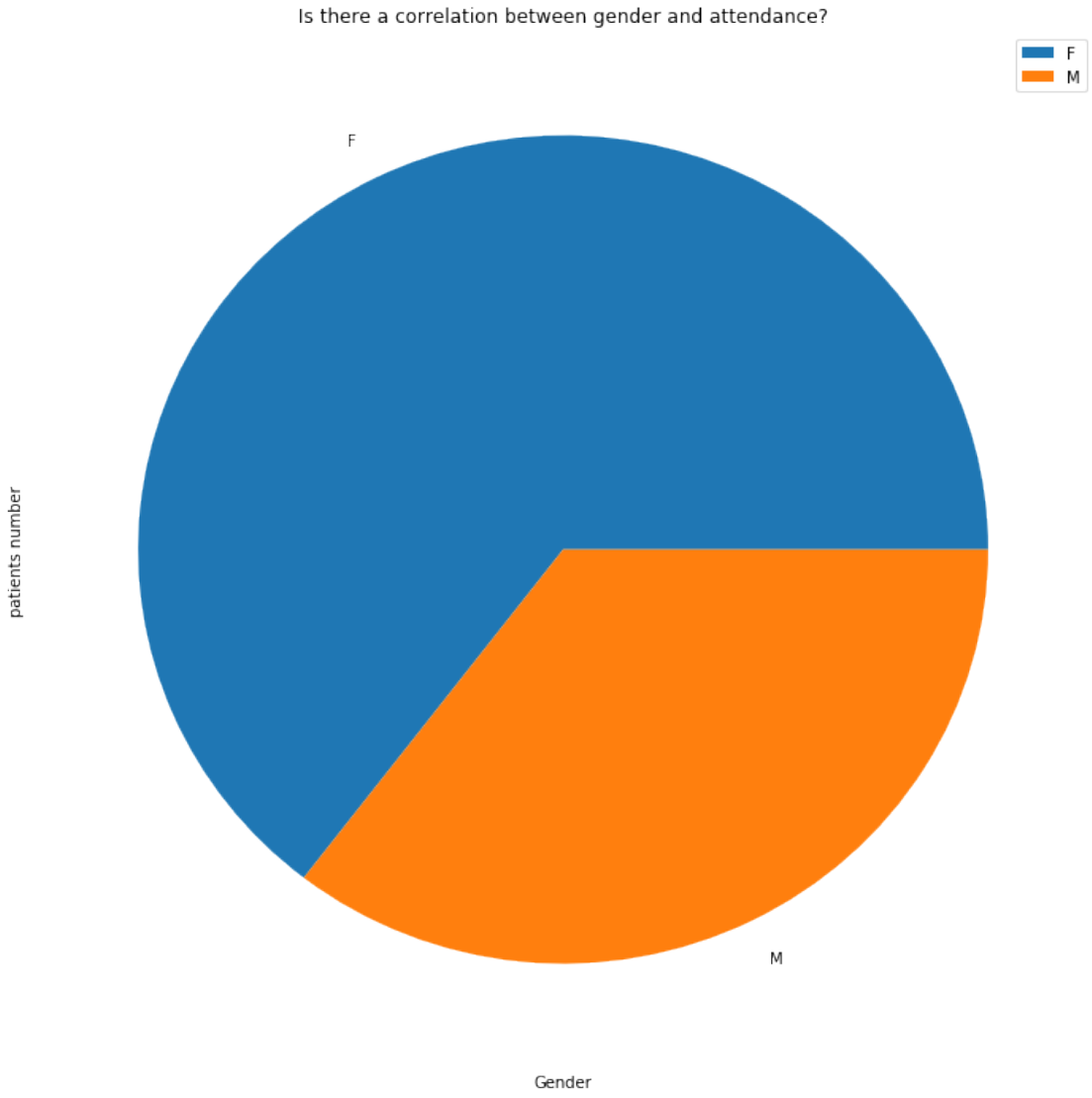
There is a clear correlation between age and attendance . After the age of 60, the number begins to decrease

```
In [21]: # Is there a correlation between illness, age and attendance?
         plt.figure(figsize=[12, 12])
         df[show].groupby(['hypertension','Diabetes']).mean()['Age'].plot(kind='bar',color='blue
         df[noshow].groupby(['hypertension','Diabetes']).mean()['Age'].plot(kind='bar',color='re
         plt.legend()
         plt.title('Is there a correlation between illness, age and attendance?')
         plt.xlabel('chronic disease')
         plt.ylabel('Age');
```

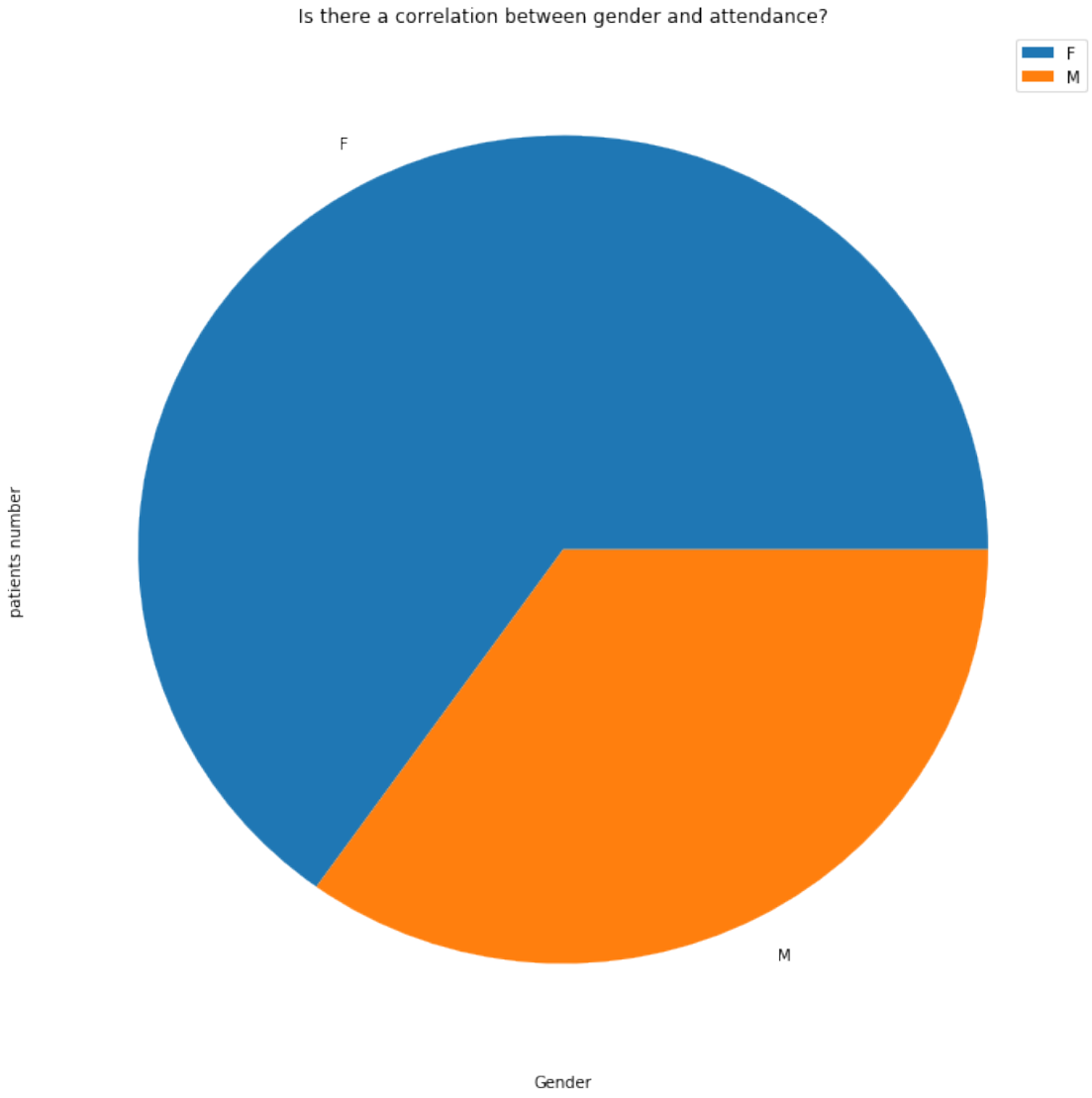


There is a correlation between age and chronic disease. As the age increases, the number of chronic disease increases, but there is no relationship between chronic disease and attendance.

```
In [22]: # Is there a correlation between gender and attendance?
         plt.figure(figsize=[12, 12])
         df.Gender[show].value_counts().plot (kind = 'pie',label = 'show')
         plt.legend()
         plt.title('Is there a correlation between gender and attendance?')
         plt.xlabel('Gender')
         plt.ylabel('patients number');
```

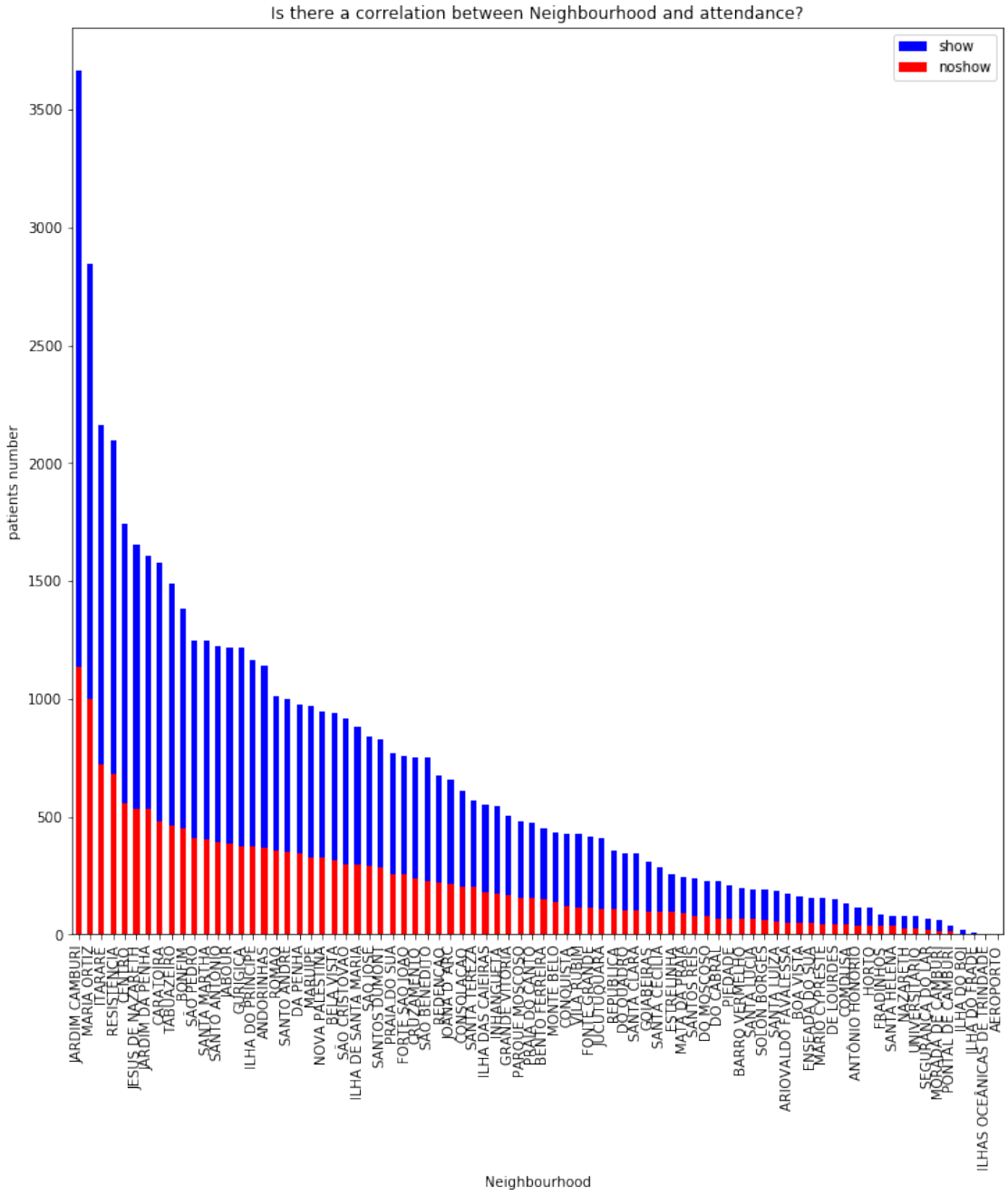



```
In [23]: plt.figure(figsize=[12, 12])
         df.Gender[noshow].value_counts().plot(kind = 'pie',label = 'noshow')
         plt.legend()
         plt.title('Is there a correlation between gender and attendance?')
         plt.xlabel('Gender')
         plt.ylabel('patients number');
```



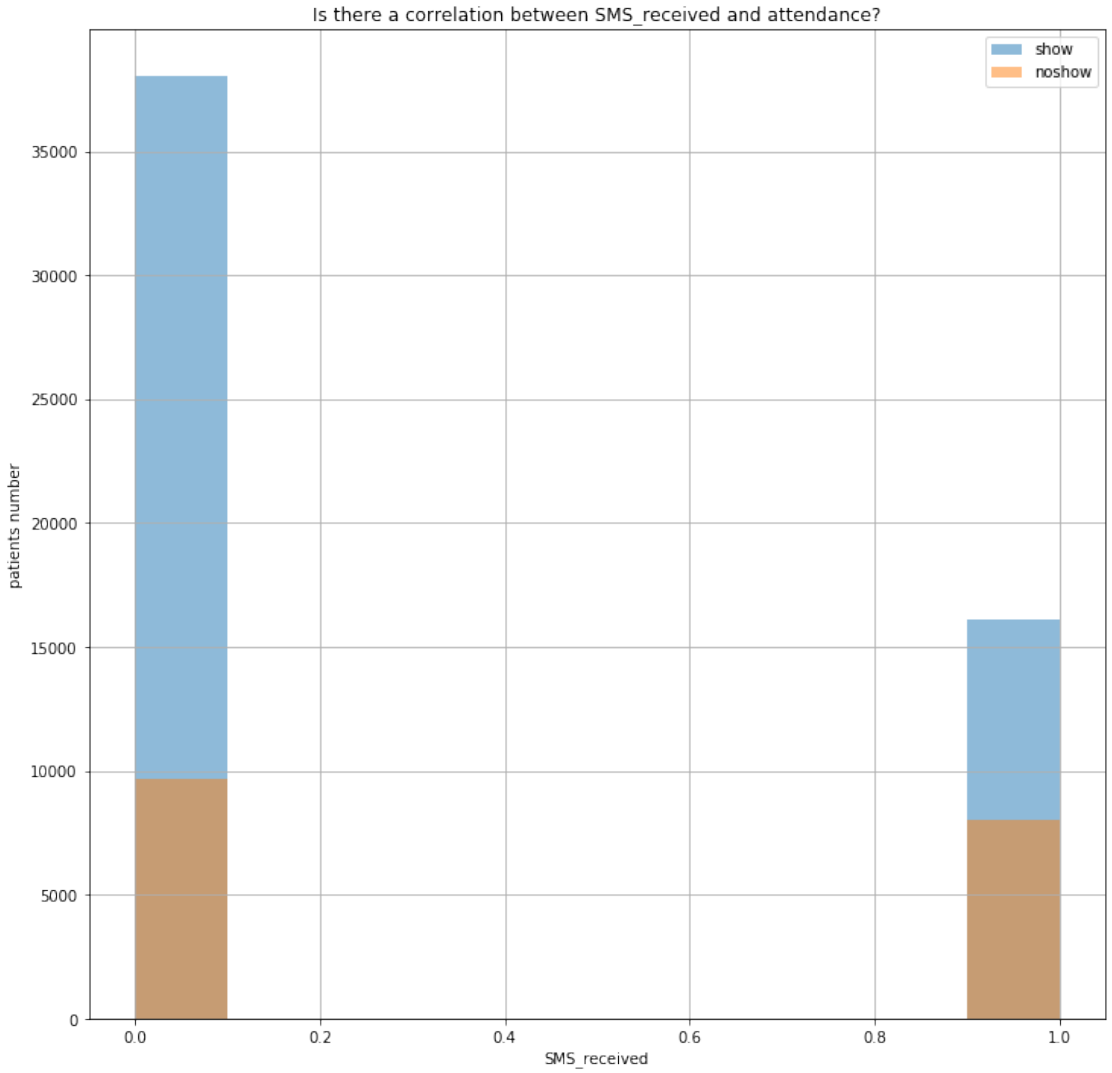
Gender does not affect attendance

```
In [24]: plt.figure(figsize=[12, 12])
         df.Neighbourhood[show].value_counts().plot(kind = 'bar',color='blue',label = 'show')
         df.Neighbourhood[noshow].value_counts().plot(kind = 'bar',color='red',label = 'noshow')

         plt.legend()
         plt.title('Is there a correlation between Neighbourhood and attendance?')
         plt.xlabel('Neighbourhood')
         plt.ylabel('patients number');
```



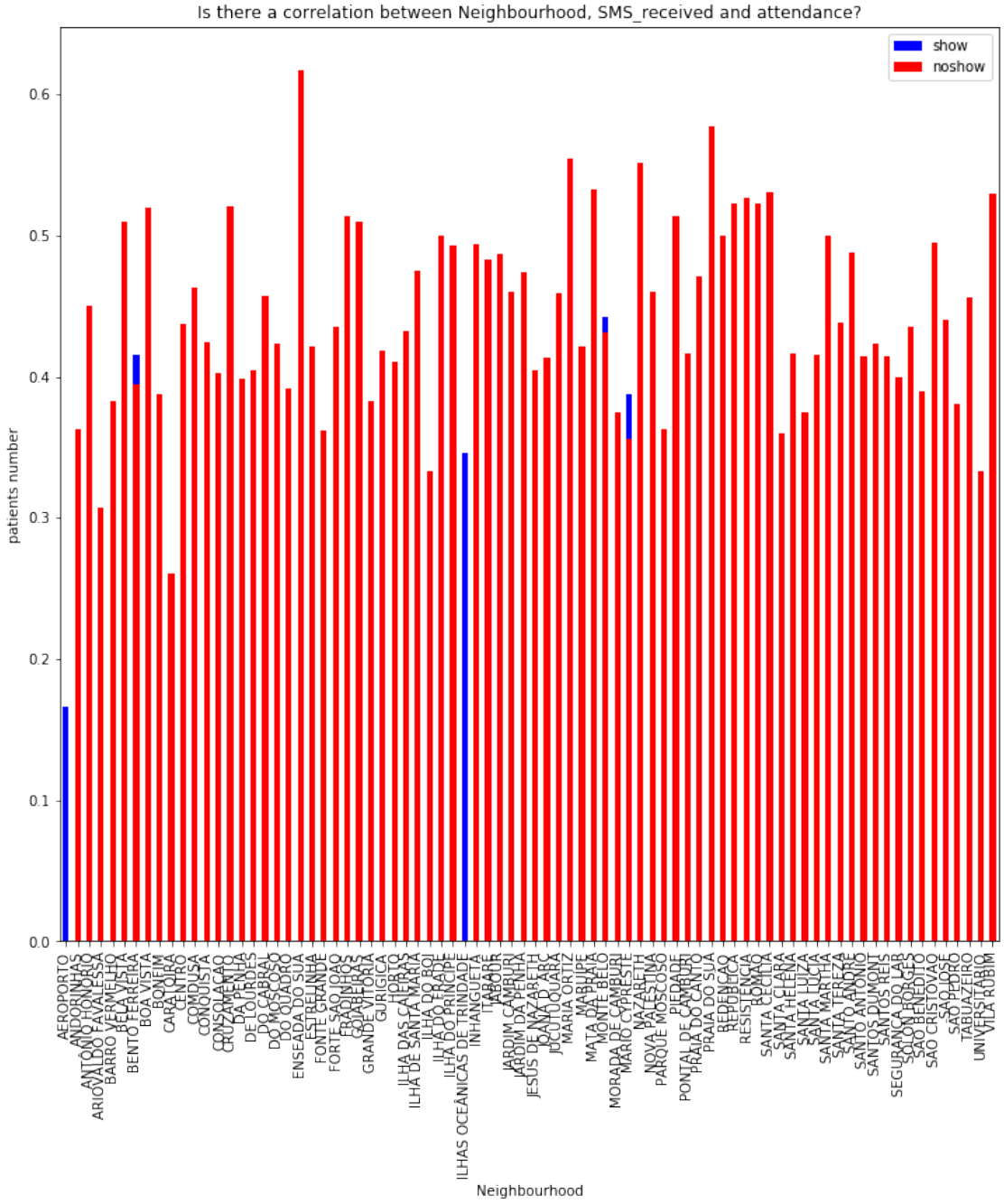
There is a clear correlation between Neighbourhood and attendance.

```
In [25]: plt.figure(figsize=[12, 12])
         df.SMS_received[show].hist(alpha=0.5,label='show')
         df.SMS_received[noshow].hist(alpha=0.5,label='noshow')
         plt.legend()
         plt.title('Is there a correlation between SMS_received and attendance?')
         plt.xlabel('SMS_received')
         plt.ylabel('patients number');
```



The number of those who attended and did not receive SMS is more than those who received and this is a strange

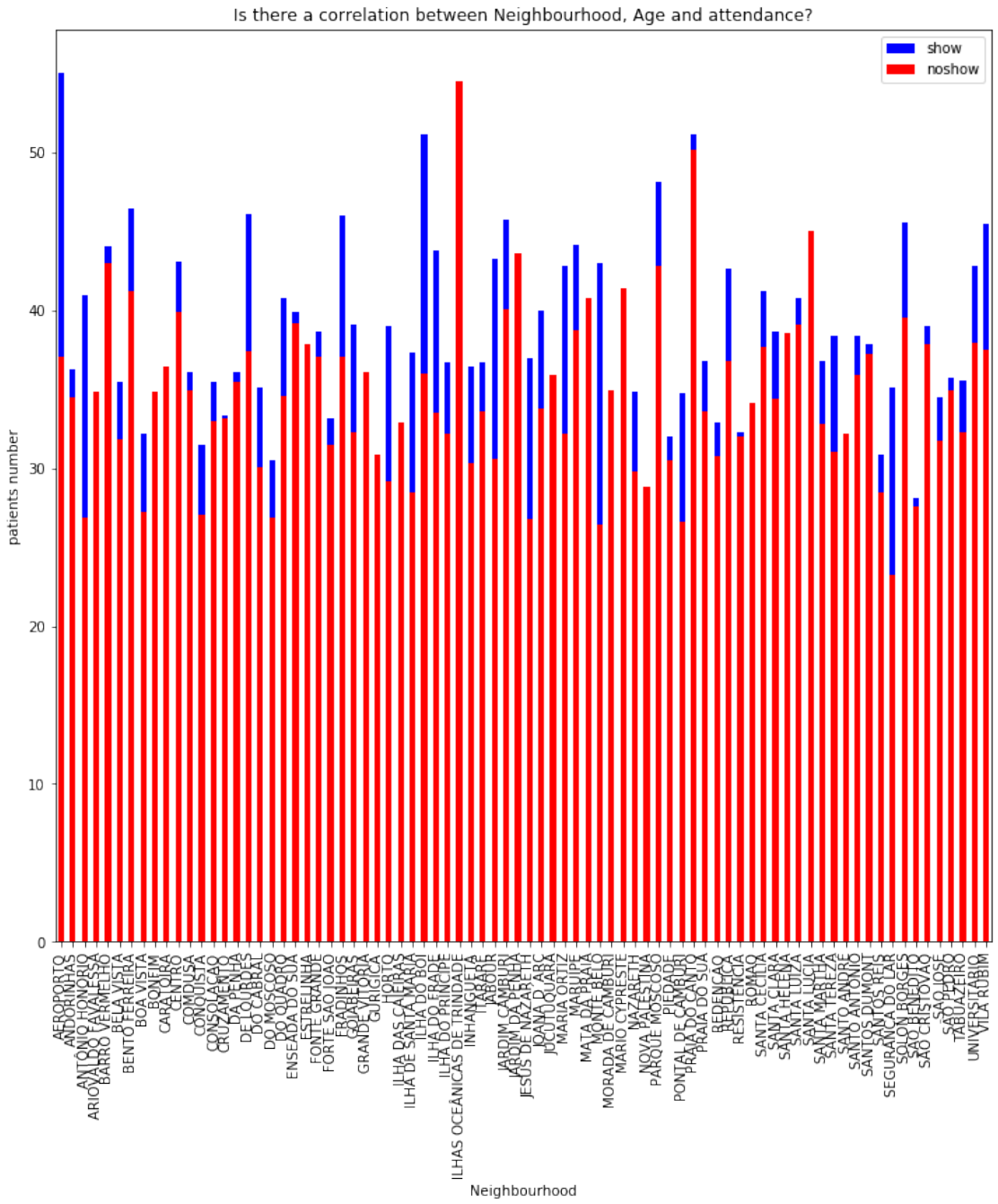
```
In [26]: # Is there a correlation between Neighbourhood, SMS_received and attendance?
         plt.figure(figsize=[12, 12])
         df[show].groupby(['Neighbourhood']).SMS_received.mean().plot(kind='bar',color='blue',la
         df[noshow].groupby(['Neighbourhood']).SMS_received.mean().plot(kind='bar',color='red',l
         plt.legend()
         plt.title('Is there a correlation between Neighbourhood, SMS_received and attendance?')
         plt.xlabel('Neighbourhood')
         plt.ylabel('patients number');
```



Yes, there is an correlation some Neighbourhood when SMS_received they all attend

```
In [27]: # Is there a correlation between Neighbourhood, Age and attendance?
         plt.figure(figsize=[12, 12])
         df[show].groupby(['Neighbourhood']).Age.mean().plot(kind='bar',color='blue',label='show
         df[noshow].groupby(['Neighbourhood']).Age.mean().plot(kind='bar',color='red',label='nos
         plt.legend()
         plt.title('Is there a correlation between Neighbourhood, Age and attendance?')
```

```
plt.xlabel('Neighbourhood')
plt.ylabel('patients number');
```



Yes, it affects some Neighbourhood whose average age is high, all of them not attended

Conclusions There is a clear correlation between age and attendance. After the age of 60, the number begins to decrease

There is a clear correlation between Neighbourhood and attendance.and affected by age and SMS_received

The number of shows who did not receive sms is more than those who did, and this is a strange thing

2.0.1 Limitations

The data did not indicate whether the patient was working or whether the working hours prevented him from attending or not

There are many features that are not useful to know if the patient will attend or not And there are negative ages, and this is illogical ## Submitting your Project

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

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

```
In [ ]:
```