Investigate_a_Dataset

January 28, 2022

1 Project: Investigate a Dataset - [No-show 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.

1.1.2 Questions for Analysis

We are going to ask some questions to understand and analyze our data like:

- Does age or gender have effect on patients attendance ?
- Does enrollement in Brasilian welfare program Bolsa Família affect showing up ?
- How can neighbourhood affect patients attendance ?
- What is the relationship between the patient's neighbourhood and receiving SMS on showing up ?
- Does the Handcap prevent patients from attending in secduled appointments ?

```
In [11]: # Use this cell to set up import statements for all of the packages that you
         #   plan to use.
         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         sns.set_style('darkgrid')
```

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

Data Wrangling

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

```
In [12]: # Loading in the data :

         df = pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
         df.head()
```

Out[12]:

	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 [13]: # Data before Cleaning
         df.shape
```

```
Out[13]: (110527, 14)
```

Our data represented in 14 columns and 110527 rows

```
In [14]: # Check missing data
         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
```

No missing data noticed

```
In [15]: # check if there are any duplicated data
         df.duplicated().sum()
```

```
Out[15]: 0
```

No duplicated data detected

```
In [16]: # More information about our data
         df.describe()
```

```
Out[16]:           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
```

Minimum value of Ages is negative which does't make sense

```
In [17]: # Checking if there are any other negative values
         (df<0).any()
```

```
Out[17]: PatientId         False
         AppointmentID     False
         Gender             True
         ScheduledDay       True
         AppointmentDay     True
         Age                True
         Neighbourhood      True
         Scholarship       False
         Hipertension      False
         Diabetes          False
         Alcoholism        False
         Handcap           False
         SMS_received      False
         No-show            True
         dtype: bool
```

Since "Gender", "ScheduledDay", "AppointmentDay", "Neighbourhood","No-show" do not have (int) or (float) values
so we only have negative vlaues in "Age" column.

```
In [18]: # Checking the negative values in "Age" column
         mask = df.query('Age < 0 ')
         mask
```

```
Out[18]:           PatientId  AppointmentID Gender          ScheduledDay  \
         99832  4.659432e+14        5775010      F  2016-06-06T08:58:13Z

                     AppointmentDay  Age Neighbourhood  Scholarship  Hipertension  \
         99832  2016-06-06T00:00:00Z   -1         ROMÃO            0             0
```

```
       Diabetes  Alcoholism  Handcap  SMS_received No-show
99832         0           0        0             0      No
```

We have only one negative value which is neglected but we gonna delete it

1.1.3 Data Cleaning

In [19]:
```
# "PatientId", "AppointmentID", "ScheduledDay", "AppointmentDay" are not important fact
## will show up for their scheduled appointment, so we gonna remove them.
df.drop(['PatientId', 'AppointmentID', 'ScheduledDay', 'AppointmentDay'], axis = 1 , in
df.head()
```

Out[19]:
```
  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
```

In [20]:
```
# Correcting data:

df.rename(columns={'Hipertension' : 'Hypertension'}, inplace = True )
df.rename(columns={'Handcap' : 'Handicap'}, inplace = True )
df.rename(columns={'No-show' : 'No_show'}, inplace = True )
df.head()
```

Out[20]:
```
  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      No
1           0         0             0      No
2           0         0             0      No
3           0         0             0      No
4           0         0             0      No
```

In [8]:
```
# Removing the negative value in Age column
df.drop(index=99832, inplace = True )
df.describe()
```

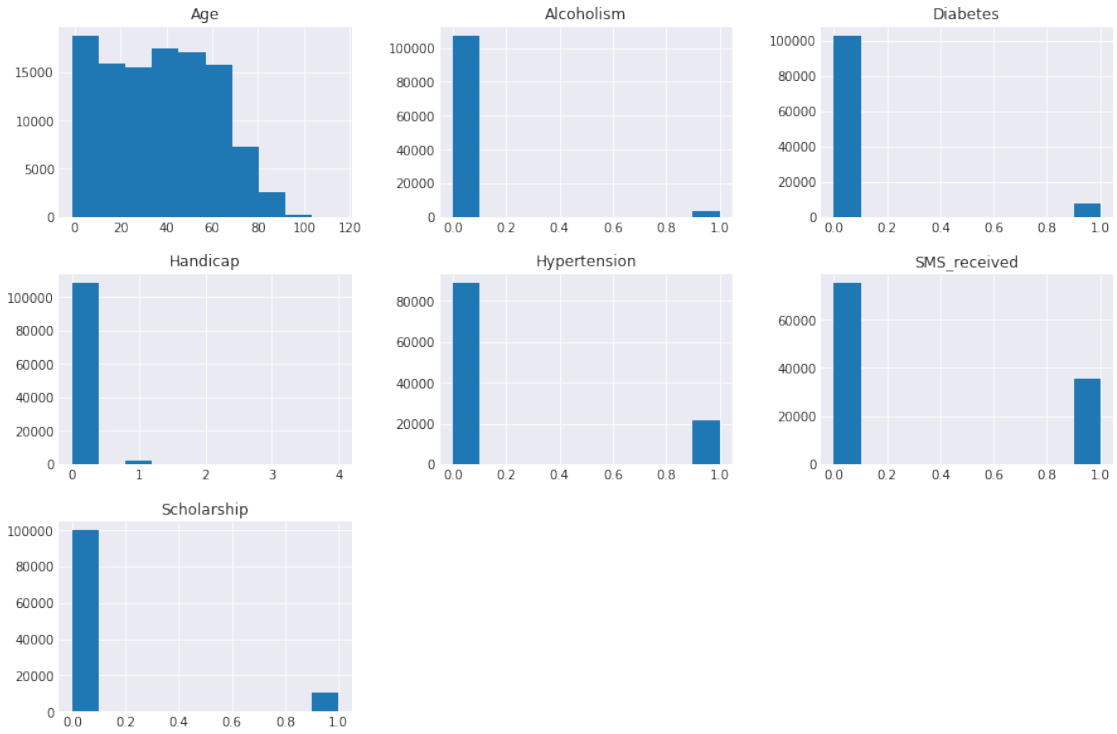
Out[8]:

	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
count	110526.000000	110526.000000	110526.000000
mean	0.030400	0.022248	0.321029
std	0.171686	0.161543	0.466874
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

Exploratory Data Analysis

In [21]:
```
# Visualizing the whole dataset:

df.hist(figsize= (15,10));
```



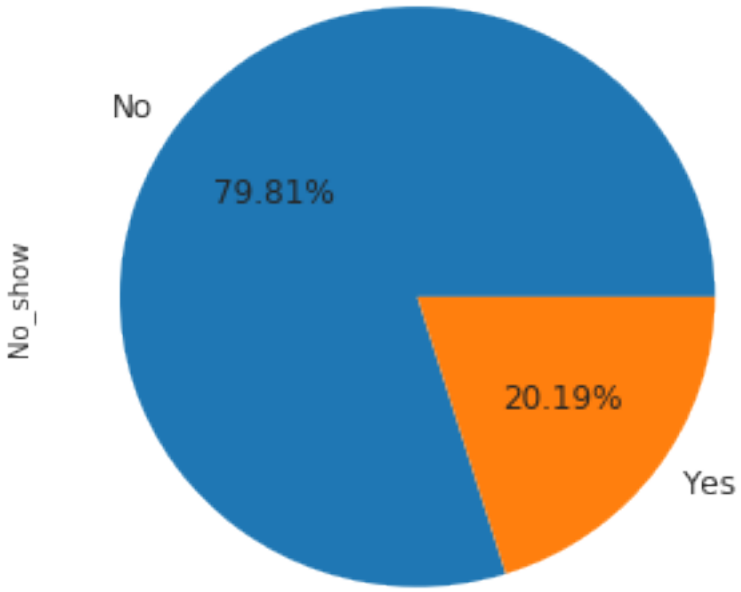
- The most common disease is Hypertension.
- About half of patients received SMS.
- Most of patiens aren't enrolled in Brasilian welfare program.
- Young patients shows up the most followed by (43-58) yers old patients.

```
In [22]: show = df.No_show == 'No'
         noshow = df.No_show == 'Yes'
         ((df[show].count() )/ (df[show].count() + df[noshow].count()))*100
```

```
Out[22]: Gender          79.806744
         Age             79.806744
         Neighbourhood   79.806744
         Scholarship     79.806744
         Hypertension    79.806744
         Diabetes        79.806744
         Alcoholism      79.806744
         Handicap        79.806744
         SMS_received    79.806744
         No_show         79.806744
         dtype: float64
```

Attendance Ratio Is : 79.8 %

```
In [14]: # Visualizing Attendance ratio:

         df['No_show'].value_counts().plot(kind='pie', figsize=(5,5),fontsize=12, autopct="%.2f%
```



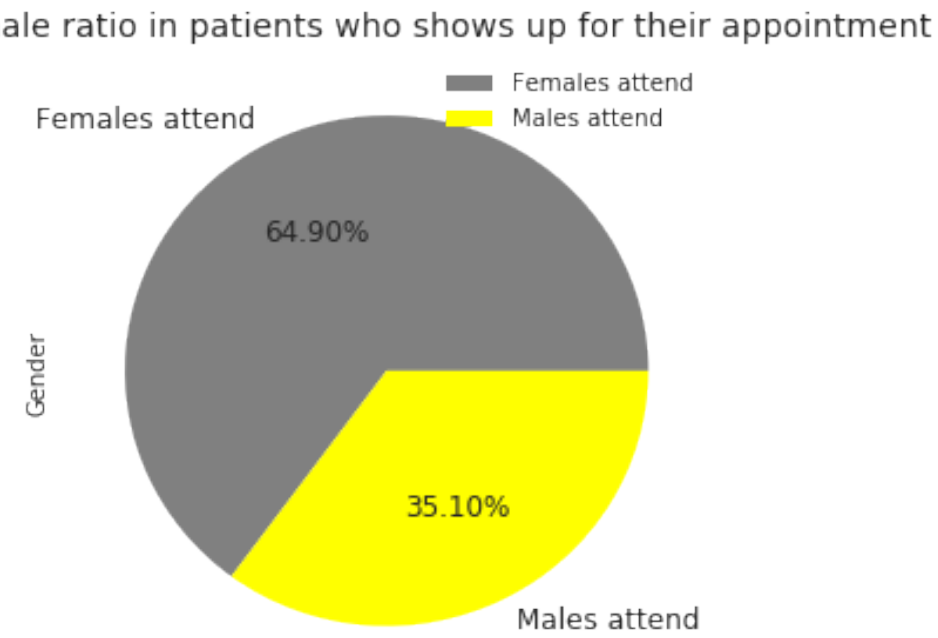
1.1.4 Question 1 (Does gender affect predicting showing up for sceduled appointment !)

In [ ]:
```
# Calculating the ratio of showing up for both males and females :

((df.Gender[show].value_counts()) / (df.Gender[show].value_counts() + df.Gender[noshow].
```

Females attendance ratio : 79.69 %
Males attendance ratio : 80.032 %
Almost the same ratio

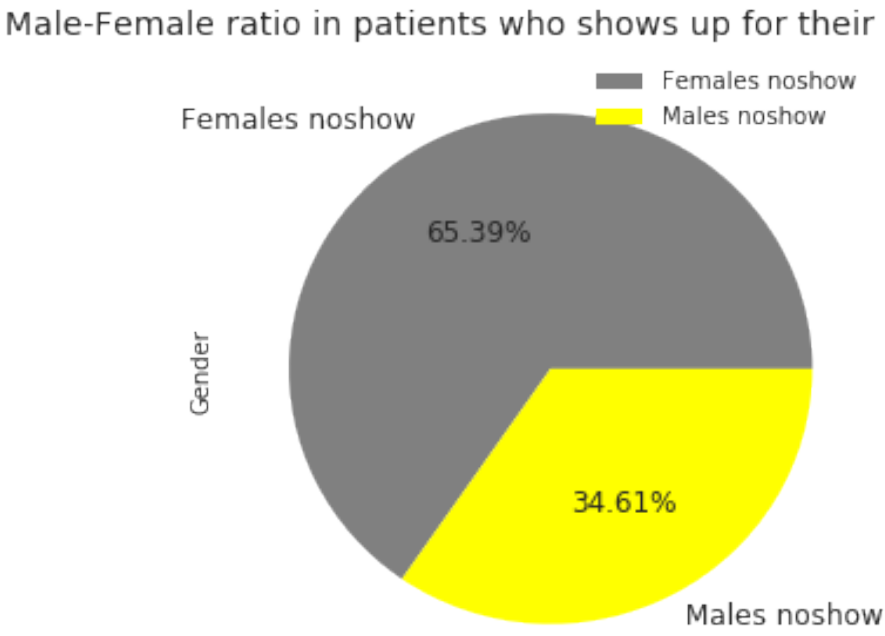
Gender is not an important factor to predict showing up for sceduled appointment

In [28]:
```
# Visualizing Male-Female ratio in patients who shows up for their appointment:


def show_ratio(factor):
    plt.figure(figsize=(5,5))
    df[factor][show].value_counts().plot(kind='pie' ,labels=['Females attend', 'Males a
    plt.legend()
    plt.title('Male-Female ratio in patients who shows up for their appointment', fonts
show_ratio("Gender")
```



In [31]:
```
# Visualizing Male-Female ratio in patients who not shows up for their appointment:
```

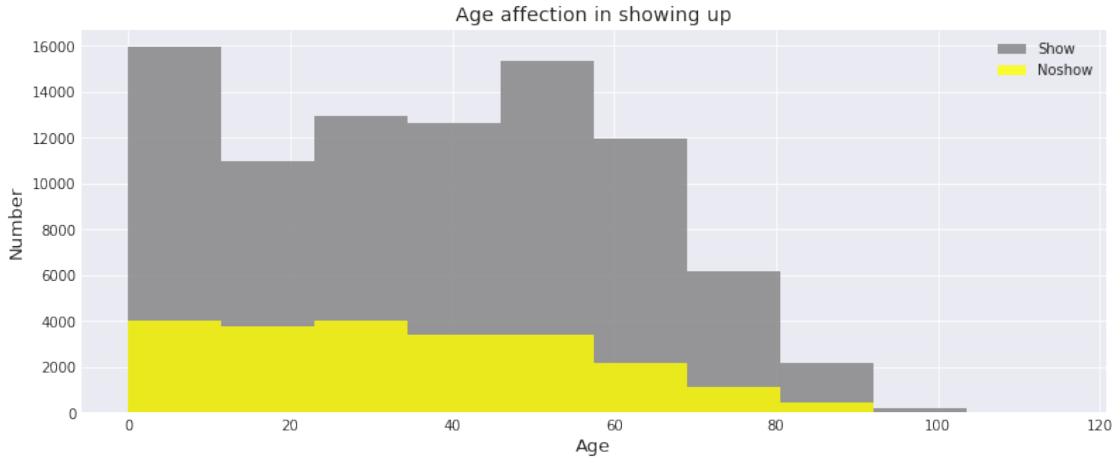
```
def show_ratio(factor):
    plt.figure(figsize=(5,5))
    df[factor][noshow].value_counts().plot(kind='pie' ,labels=['Females noshow', 'Males
    plt.legend()
    plt.title('Male-Female ratio in patients who shows up for their appointment', fonts
show_ratio("Gender")
```



As we see, The percntage of females and males in both showing up or not is nearly equal

1.1.5 Question 2 (Does Age affect predicting showing up for sceduled appointment !)

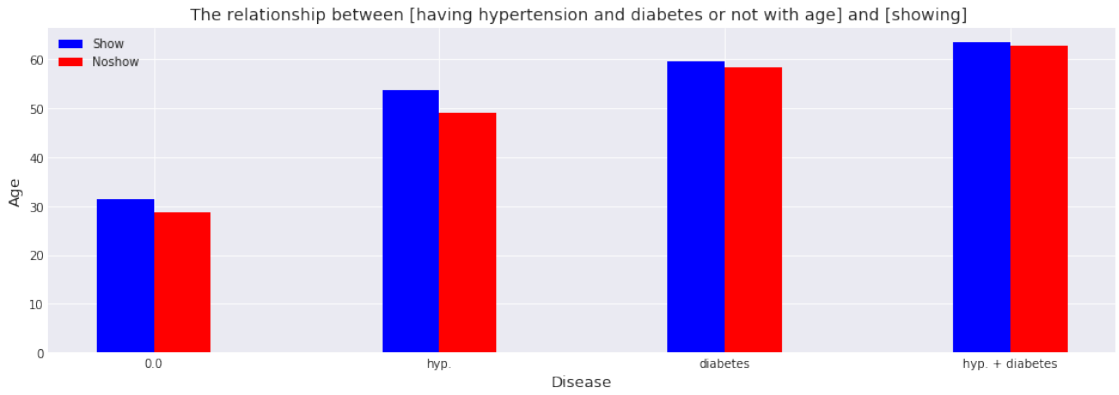
In [46]:
```
# Visualizing showing up according to patients age:

plt.figure(figsize=(13,5))
a = df.Age[show]
b = df.Age[noshow]

age_show = a.hist (color='grey',alpha=0.8, label='Show')
age_noshow = b.hist(color='yellow',alpha=0.8, label='Noshow')

plt.title('Age affection in showing up', fontsize=14 )
plt.xlabel('Age', fontsize=13)
plt.ylabel('Number', fontsize=13)
plt.legend();
```



Age affects showing up as we see - Young patients of (0-10) years are more likely to show up - Fllowed by (43-58) group - Patients older than 65 seems like they need a homecare services

1.1.6 Question 3 (What is the relationship between a specific disease with age on showing!)

```
In [68]: # Visualizing the relationship between [having hypertension and diabetes or not with ag

         group1= df[show].groupby(['Hypertension', 'Diabetes']).mean()['Age']
         group2= df[noshow].groupby(['Hypertension', 'Diabetes']).mean()['Age']

         ind = np.arange(len(group1))
         width= 0.2
         plt.figure(figsize=(16,5))
         attendance = plt.bar(ind, group1, width, color='b', label='Show')
         abscence =plt.bar(ind + width, group2 , width, color='r', label='Noshow')

         locations = ind + width / 2
         labels= ['0.0', 'hyp.','diabetes','hyp. + diabetes']
         plt.xticks(locations, labels)
         plt.title('The relationship between [having hypertension and diabetes or not with age]
         plt.ylabel('Age', fontsize=13)
         plt.xlabel('Disease', fontsize=13)

         plt.legend();
```



Having "Hypertension" or "Diabetes" or both with aging doesn't affect showing up for appoinments

1.1.7 Question 4 (Does scolarship matter !)

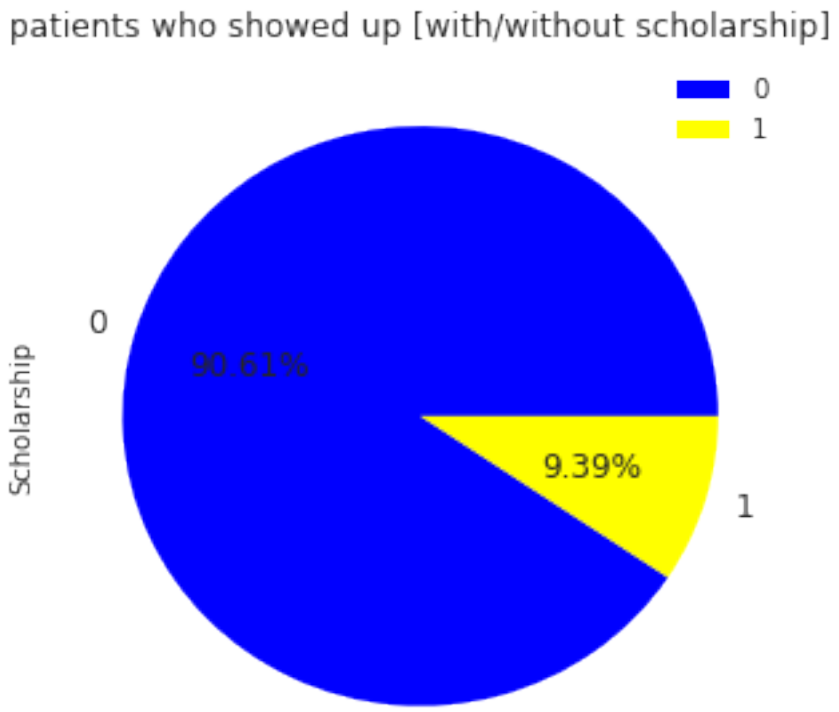
```
In [28]: # Calculating showing up ratio according to owning scholarship :

         (df.Scholarship[show].value_counts() / df.Scholarship.value_counts())*100
```

```
Out[28]: 0    80.192645
         1    76.263696
         Name: Scholarship, dtype: float64
```

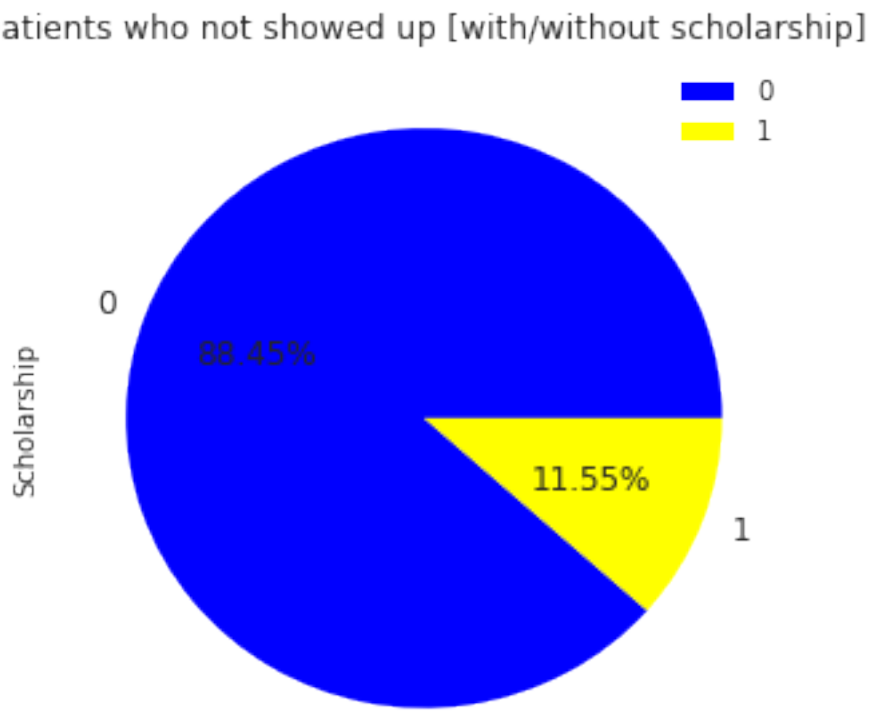
80.19 % of patients with no scolarship showed up
76.26 % of patients with scolarship showed up

Scholarship hasn't a great effect on showing up

```
In [27]: # Show Patients [with/without] enrollment in Brasil welfare program :

         def show_ratio(factor):
             plt.figure(figsize=(5,5))
             df[factor][show].value_counts().plot(kind='pie' , colors = ['blue', 'yellow'], font
             plt.legend()
             plt.title('patients who showed up [with/without scholarship]', fontsize=12);
         show_ratio("Scholarship")
```



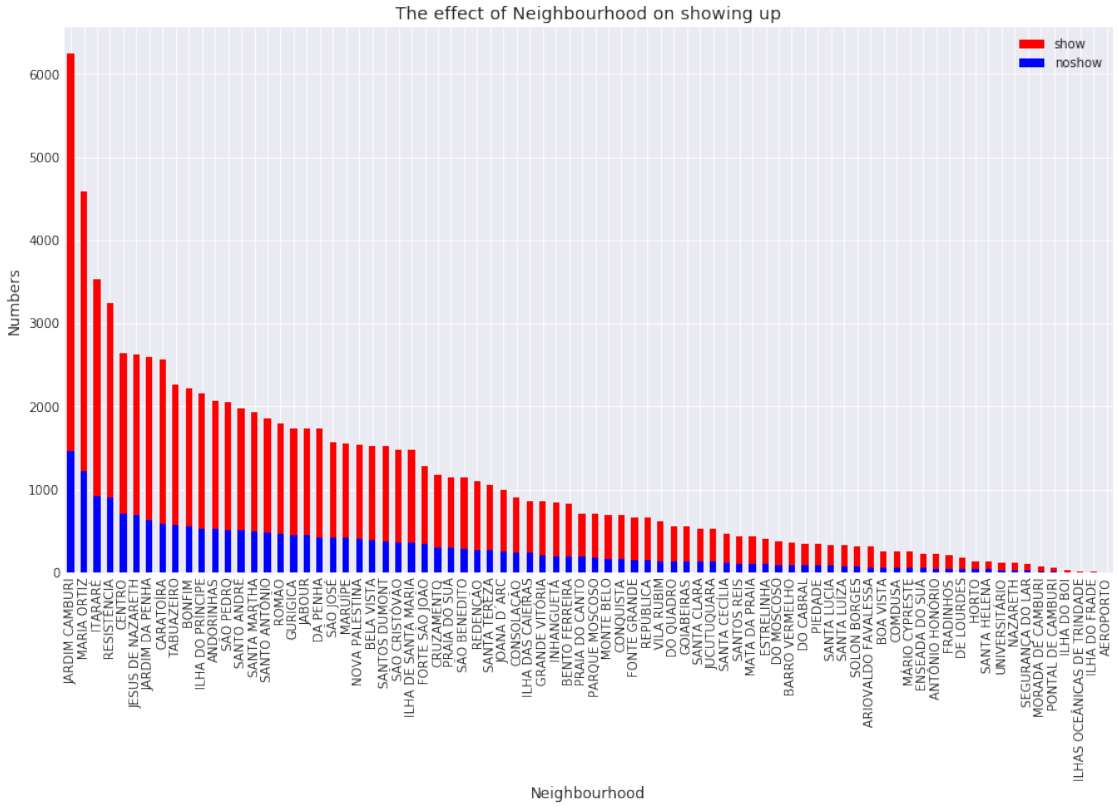
In [26]:
```
# NoShow Patients [with/without] enrollment in Brasil welfare program :

def show_ratio(factor):
    plt.figure(figsize=(5,5))
    df[factor][noshow].value_counts().plot(kind='pie' , colors = ['blue', 'yellow'], fo
    plt.legend()
    plt.title('patients who not showed up [with/without scholarship]', fontsize=12);
show_ratio("Scholarship")
```



The ratio is so close , so enrolling in Brasil welfare program is not a factor to predict showing up for appointments

1.1.8 Question 5 (Does Neighbourhood affect showing up !)

In [84]:
```
# Visualizing the effect of Neighbourhood on showing up:

plt.figure(figsize = [15,8])
show_data = df.Neighbourhood[show].value_counts()
noshow_data = df.Neighbourhood[noshow].value_counts()

neighbourhood_show = show_data.plot( kind= 'bar', color= 'red' , label = 'show' )
neighbourhood_noshow= noshow_data.plot(kind= 'bar', color= 'blue' , label = 'noshow' )
plt.legend()
plt.title('The effect of Neighbourhood on showing up', fontsize=14)
plt.xlabel('Neighbourhood', fontsize=12)
plt.ylabel('Numbers', fontsize=12);
```



Neibourhood has a great effect on showing up

1.1.9 Question 6 (Does SMS make deffirence in pateints attendance !)

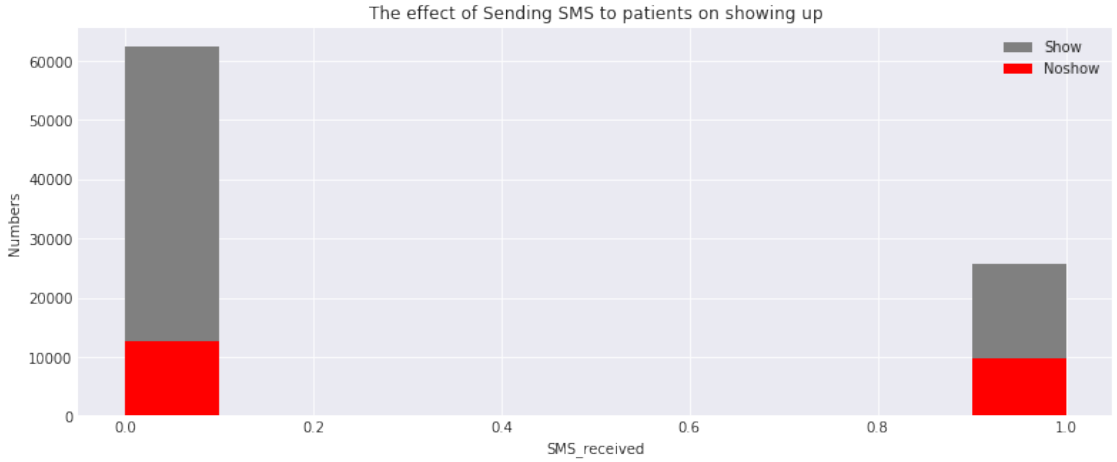
```
In [14]: ( df.SMS_received[show].value_counts() / df.SMS_received.value_counts() )*100
```

```
Out[14]: 0    83.296466
         1    72.425455
         Name: SMS_received, dtype: float64
```

Percentage of patients who showed up without receiving SMS = 83.296 %
Precentage of patients who showed up and recevied SMS = 72.425 %

```
In [12]: # Visualizing SMS_received data:

         plt.figure(figsize = [13,5])
         a= df.SMS_received[show]
         b= df.SMS_received[noshow]

         sms_show = a.hist(color='grey', label='Show')
         sms_noshow = b.hist(color='red', label='Noshow')

         plt.legend()
```

```
plt.title('The effect of Sending SMS to patients on showing up')
plt.xlabel('SMS_received')
plt.ylabel('Numbers');
```



Sending SMS doesn't matter , because the number of patients showed up without receiving SMS is greater than those who received one.

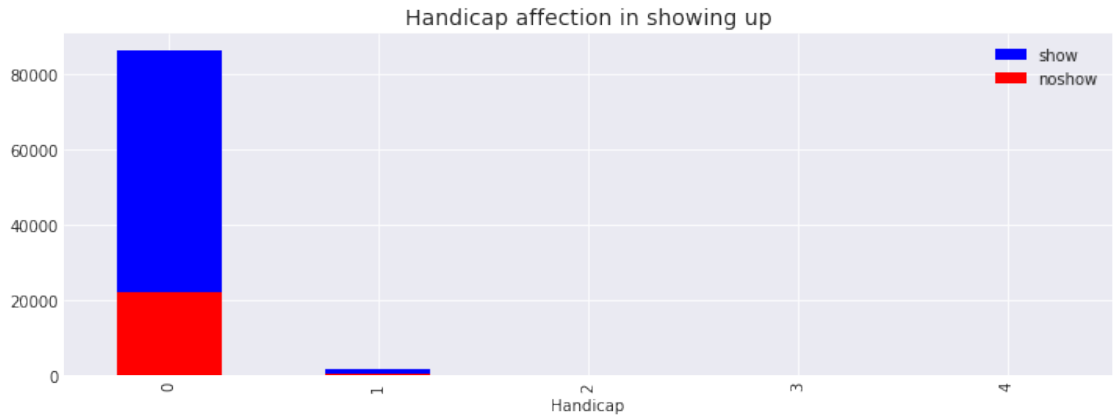
1.1.10 Question 7 (Does handicap affect attendance in time !)

```
In [21]: # Calculating the effect of handcap on attendance :

         ( df.Handcap[show].value_counts() / df.Handcap.value_counts() ) * 100
```

```
Out[21]: 0    79.764510
         1    82.076396
         2    79.781421
         3    76.923077
         4    66.666667
         Name: Handcap, dtype: float64
```

Handcap affect showing up in time , and type four has the lowest attendance ratio

```
In [15]: # Visualizing Hancap data:

         a= df.Handicap[show].value_counts()
         b= df.Handicap[noshow].value_counts()

         handicap_show = a.plot(kind='bar' , color = 'blue', label = 'show', figsize=(12,4))
         handicap_noshow= b.plot(kind='bar' ,color = 'red', label = 'noshow' ,figsize=(12,4))
         plt.title('Handicap affection in showing up', fontsize= 14)
         plt.legend()
         plt.xlabel('Handicap');
```



Conclusions

Older patients with hypertension and diabetes don't show up as much as the younger patients.

Sending SMS doesn't matter , because the number of patients showed up without receiving SMS is greater than those who received one and it only effictive in 4 neighbourhood.

Handcap affect showing up in time but not very much, and type four has the lowest attendance ratio.

Neibourhood has a great effect on showing up.

Age has an important effect on showing , the older the patient is the less he shows up and young patients (0-10) shows up the most , followed by the age of (43-58).

1.2 Limitations

SMS , Gender , Brasilian welfare program enrollment and having Hypertension or diabetes are not important facors predicting showing up.

```
In [32]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[32]: 0
```

```
In [ ]:
```