investigate-a-dataset-template

November 18, 2020

1 Project: No Show Appointments DataSet.

1.1 Table of Contents

Introduction

Data Wrangling

Exploratory Data Analysis

Conclusions

Introduction

The Medical Appointment No Shows is selected for the analysis. This dataframe is consisted of 14 variables : 01 - PatientId Identification of a patient 02 - AppointmentID Identification of each appointment 03 - Gender Male or Female . Female is the greater proportion, woman takes way more care of they health in comparison to man. 04 - DataMarcacaoConsulta The day of the actuall appointment, when they have to visit the doctor. 05 - DataAgendamento The day someone called or registered the appointment, this is before appointment of course. 06 - Age How old is the patient. 07 - Neighbourhood Where the appointment takes place. 08 - Scholarship True of False . Observation, this is a broad topic, consider reading this article 09 - Hipertension True or False 10 - Diabetes True or False 11 - Alcoholism True or False 12 - Handcap True or False 13 - SMS_received 1 or more messages sent to the patient. 14 - No-show True or False.

Three quastions will be asked: Question 1 Who tend to show to thier Appointments young people or old people?

Question 2 Who didn't show more to their appointment patients with hipertension, diabetes, alcoholism, or handcap?

Question 3 Does having a scholarship have an effect if a patient shows or doesn't show to his appointment?

[1]:
```
# importing important libraries.
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

Data Wrangling

1.1.1 General Properties

[2]:
```
df = pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
df.head(8)
```

[2]:

	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	
5	9.598513e+13	5626772	F	2016-04-27T08:36:51Z	
6	7.336882e+14	5630279	F	2016-04-27T15:05:12Z	
7	3.449833e+12	5630575	F	2016-04-27T15:39:58Z	

	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	
5	2016-04-29T00:00:00Z	76	REPÚBLICA	0	1	
6	2016-04-29T00:00:00Z	23	GOIABEIRAS	0	0	
7	2016-04-29T00:00:00Z	39	GOIABEIRAS	0	0	

	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
5	0	0	0	0	No
6	0	0	0	0	Yes
7	0	0	0	0	Yes

The name of the last column should be changed to `NoShow` for less complexity.

[3]:
```
#the dimensions of the dataframe
df.shape
```

[3]:
```
(110527, 14)
```

we have 14 variables and 110,527 cases (appointments).

[4]:
```
#Summary of the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64  
 2   Gender          110527 non-null  object 
 3   ScheduledDay    110527 non-null  object 
 4   AppointmentDay  110527 non-null  object 
 5   Age             110527 non-null  int64  
 6   Neighbourhood   110527 non-null  object 
 7   Scholarship     110527 non-null  int64  
 8   Hipertension    110527 non-null  int64  
 9   Diabetes        110527 non-null  int64  
 10  Alcoholism      110527 non-null  int64  
 11  Handcap         110527 non-null  int64  
 12  SMS_received    110527 non-null  int64  
 13  No-show         110527 non-null  object 
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

we can see that this dataframe has no missing values - luckly - the `SceduledDay` and `AppointmentDay` are object instead of date.

[5]:
```
# to see if this dataframe has duplicated values
sum(df.duplicated())
```

[5]: 0

NO duplicated values in this dataframe

[6]:
```
#describtive statics
df.describe()
```

[6]:

	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
```

Clearly there is a wrong data entry in Age, where min value = -1, and max = 115 it might be propable but not very likely!

Handcap max = 4, but it should be only 0 and 1 (False or True)

[7]:
```
#check these incorrect data
df[df['Age'] == -1]
```

[7]:
```
          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
```

[8]:
```
df[df['Age'] == 115]
```

[8]:
```
          PatientId  AppointmentID Gender          ScheduledDay  \
63912  3.196321e+13        5700278      F  2016-05-16T09:17:44Z
63915  3.196321e+13        5700279      F  2016-05-16T09:17:44Z
68127  3.196321e+13        5562812      F  2016-04-08T14:29:17Z
76284  3.196321e+13        5744037      F  2016-05-30T09:44:51Z
97666  7.482346e+14        5717451      F  2016-05-19T07:57:56Z

             AppointmentDay  Age Neighbourhood  Scholarship  Hipertension  \
63912  2016-05-19T00:00:00Z  115    ANDORINHAS            0             0
```

```
63915  2016-05-19T00:00:00Z  115    ANDORINHAS            0             0
68127  2016-05-16T00:00:00Z  115    ANDORINHAS            0             0
76284  2016-05-30T00:00:00Z  115    ANDORINHAS            0             0
97666  2016-06-03T00:00:00Z  115      SÃO JOSÉ            0             1

       Diabetes  Alcoholism  Handcap  SMS_received No-show
63912         0           0        1             0     Yes
63915         0           0        1             0     Yes
68127         0           0        1             0     Yes
76284         0           0        1             0      No
97666         0           0        0             1      No
```

Here it shows that same person made 4 appointments and only went to the last one of them. As for the `Age` change age = -1 and age =115.

For the handcap value we must check if there are other values besides 4.

[9]:
```
df.Handcap.value_counts()
```

[9]:
```
0    108286
1      2042
2       183
3        13
4         3
Name: Handcap, dtype: int64
```

we can clearly see that there is values of 2 and 3, but there should be only 0 and 1 for Handcap values hold true or false only as it is described here. We should change 2, 3, and 4 values to 1 as they hold the true value.

[ ]:
```
```

1.1.2 Data Cleaning

Cleaning process

First: Change column name from no-show to NoShow

[10]:
```
df.rename(columns={'No-show': 'NoShow'}, inplace = True)
```

Second: Change Day Variables type from object to datetime

[11]:
```
#changing the ScheduledDay to datetime type
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
```

[12]:
```
#changing the AppointmentDay to datetime type
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])
```

[13]:
```
#use the info() function to check our changes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64
 2   Gender          110527 non-null  object
 3   ScheduledDay    110527 non-null  datetime64[ns, UTC]
 4   AppointmentDay  110527 non-null  datetime64[ns, UTC]
 5   Age             110527 non-null  int64
 6   Neighbourhood   110527 non-null  object
 7   Scholarship     110527 non-null  int64
 8   Hipertension    110527 non-null  int64
 9   Diabetes        110527 non-null  int64
 10  Alcoholism      110527 non-null  int64
 11  Handcap         110527 non-null  int64
 12  SMS_received    110527 non-null  int64
 13  NoShow          110527 non-null  object
dtypes: datetime64[ns, UTC](2), float64(1), int64(8), object(3)
memory usage: 11.8+ MB
```

Third: Change the incorrect ages.

[14]:
```
#change Age = -1 to 1 using replace method
df.Age = df.Age.replace(-1, 1)
```

[15]:
```
# see if our cleaning worked
df[df['Age'] == -1]
```

[15]:
```
Empty DataFrame
Columns: [PatientId, AppointmentID, Gender, ScheduledDay, AppointmentDay, Age,
Neighbourhood, Scholarship, Hipertension, Diabetes, Alcoholism, Handcap,
SMS_received, NoShow]
Index: []
```

As for the entry of `Age` = 115, I will assume it's a entry process mistake and change it to 15

[16]:
```
#change Age = 115 to Age = 15
df.Age = df.Age.replace(115, 15)
```

[17]:
```
#check if it did change
df.describe()
```

[17]:
```
          PatientId  AppointmentID            Age    Scholarship  \
count  1.105270e+05   1.105270e+05  110527.000000  110527.000000
mean   1.474963e+14   5.675305e+06      37.084369       0.098266
std    2.560949e+14   7.129575e+04      23.104711       0.297675
min    3.921784e+04   5.030230e+06       0.000000       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     102.000000       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
```

Age min = 0, we can see that Age = -1 was corrected as for min = 0 it possibly means an infant, and for max = 102 it's an accepted value as an age for person.

Fourth: change `HandCap` = (4, 3, 2) to 1, as it only accepts 0 or 1.

[18]:
```
#Using replace method to change Handcap values to 1, but we will be using a for␣
↪loop

for n in range(2,5):
    df.Handcap = df.Handcap.replace(n, 1)
```

[19]:
```
#Test our changes
df.Handcap.unique()
```

[19]:
```
array([0, 1])
```

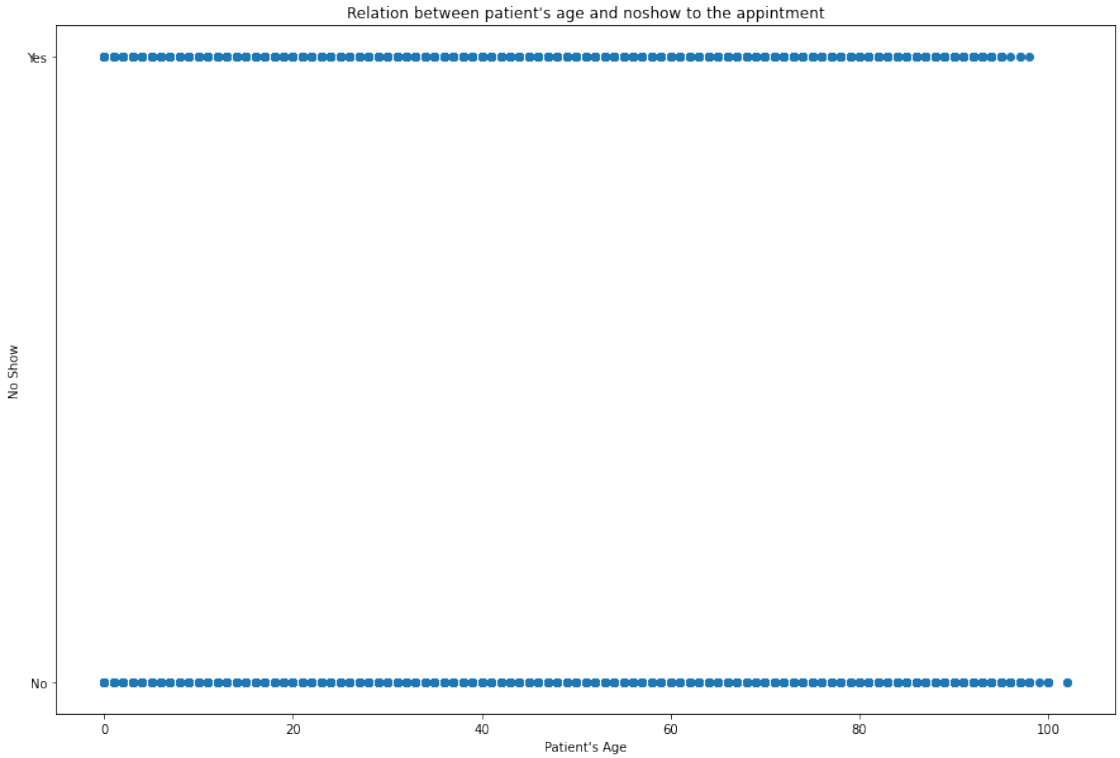
Exploratory Data Analysis

1.1.3 Question 1 Does the age of the a patients have an effect on showing to his appointment or not?

Here we want to find if there's a correlation between patient's age and NoShow variable. First: use a plot to investigate the relation ship between `Age` and `NoShow`.

[20]:
```
plt.figure(figsize=[15,10])
plt.scatter(df.Age, df.NoShow)
plt.xlabel("Patient's Age")
plt.ylabel('No Show')
plt.title("Relation between patient's age and noshow to the appintment")
```

[20]:
```
Text(0.5, 1.0, "Relation between patient's age and noshow to the appintment")
```



This kind of plot can't tell the relation between `Age` and `NoShow`. Second: divide the `Age` to two categories Age < 40 and Age $>= 40$ and try to find a correlation.

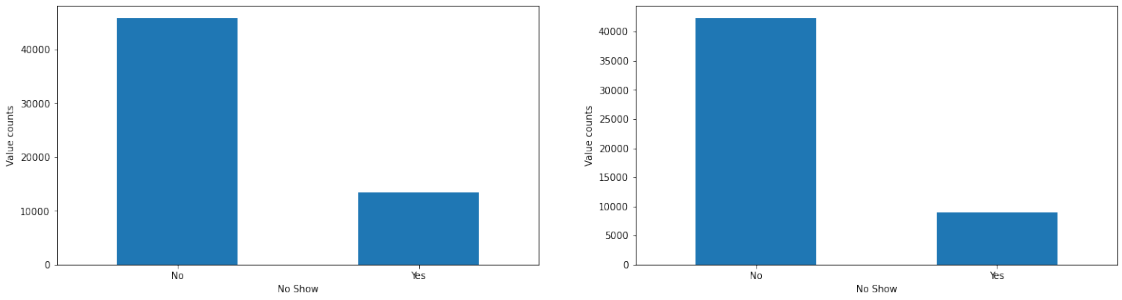
[21]:
```
#divide dataframe to two dataframes young and old
df_young = df[df['Age'] < 40]
df_old = df[df['Age'] >= 40]
```

[22]:
```
#plot for comparsion
plt.figure(figsize=[20,5])

#1st plot for df_young
plt.subplot(1, 2, 1)
df_young['NoShow'].value_counts().plot(kind = 'bar')
plt.xlabel('No Show')
plt.ylabel('Value counts')
plt.xticks(rotation = 0)
#plt.legend('Age <= 40')

#2nd plot for df_old
plt.subplot(1, 2, 2)
df_old['NoShow'].value_counts().plot(kind = 'bar')
plt.xlabel('No Show')
plt.ylabel('Value counts')
plt.xticks(rotation = 0)
#plt.legend('Age > 40')
```

[22]:
```
(array([0, 1]), [Text(0, 0, 'No'), Text(1, 0, 'Yes')])
```



People with age less than 40 show more in their appointments, but they also do not show in their appointment more than elder people, and this maybe because the number of younger people is more than elder. Using the mean will give a more specific details for the relation between age and noshow.

Calculating the mean for **Age** younger than 40 and **Age** equal or older than 40 who didn't show manually.

[23]:
```
df.query('Age < 40')['NoShow'].value_counts().No, df.query('Age <␣
↪40')['NoShow'].value_counts().Yes
```

[23]:
```
(45863, 13424)
```

[24]:
```
#mean of young patients (Age < 40) who didn't show to overall who didn't show
```

```
mean_young_noshow = (df.query('Age < 40')['NoShow'].value_counts().Yes)/(df.
↪query('Age')['NoShow'].value_counts().Yes)
mean_young_noshow
```

[24]: 0.5398101978446196

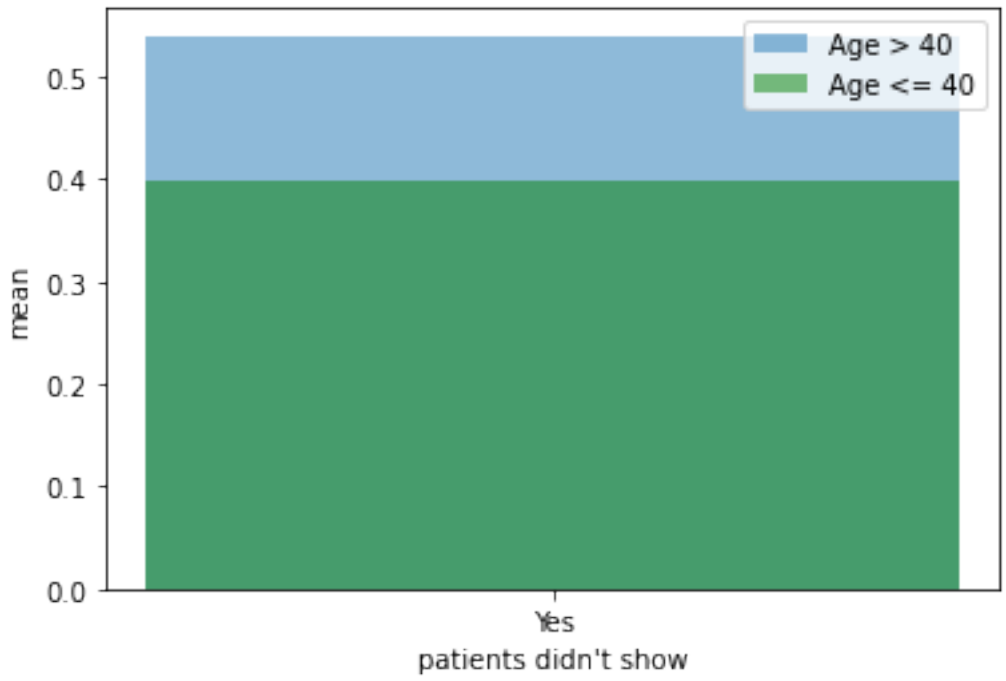
[25]:
```
#mean of old patients (Age >= 40) who didn't show to overall who didn't show
mean_old_noshow = (df.query('Age >= 40')['NoShow'].value_counts().Yes)/(df.
↪query('Age >= 40')['NoShow'].value_counts().Yes + df.query('Age <␣
↪40')['NoShow'].value_counts().Yes )
mean_old_noshow
```

[25]: 0.3985393610824858

[26]:
```
df.query('Age < 40')['NoShow'].unique()[1]
```

[26]: 'Yes'

[27]:
```
#plot tha calculated mean
plt.bar(df.query('Age < 40')['NoShow'].unique()[1], mean_young_noshow, alpha =␣
↪0.5, label = 'Age > 40')
plt.bar(df.query('Age >= 40')['NoShow'].unique()[1], mean_old_noshow, alpha = 0.
↪5, color = 'green', label = 'Age <= 40')
plt.xlabel("patients didn't show")
plt.ylabel('mean')
plt.legend()
plt.show();
```



[28]:
```
df.query('Age >= 40')['NoShow'].value_counts().No, df.query('Age >=␣
↪40')['NoShow'].value_counts().Yes
```

[28]:
```
(42345, 8895)
```

patients with Age >= 40 show to their appointments with a 14% higher than patients with Age < 40.

Third: Divide the Ages to range = 10 years, to have more detailes about the relation between Age and NoShow. 1st range (Age < 11), 2nd range (10 < Age < 21), 3rd (20 < Age < 31), 4th range (30 < Age < 41), 5th range (40 < Age < 51), 6th range (50 < Age < 61), 7th range (60 < Age < 71), 8th range (70 < Age < 81), 9th range (80 < Age < 91), 10th range (90 < Age < 101), 11th range (100 < Age < 111).

Calculating the mean for each to over all noshow:

1- create a function to calculate the mean for each age range.

2- plot the age range vs. age mean.

[38]:
```
# function for calaculating the mean
def age_range_mean(range_number, range_upper_value, range_lower_value = 'Age <␣
↪0'):

    for i in range(range_number):
        if range_lower_value == 'Age < 0':
            mean = (df.query(range_upper_value)['NoShow'].value_counts().Yes/␣
↪df.query('Age')['NoShow'].value_counts().Yes)

        else:
            mean = (df.query(range_upper_value)['NoShow'].value_counts().Yes -␣
↪df.query(range_lower_value)['NoShow'].value_counts().Yes) / df.
↪query('Age')['NoShow'].value_counts().Yes

    return mean

#test
mean_1st_range = age_range_mean(1, 'Age <11')
mean_2nd_range = age_range_mean(2, 'Age <21', 'Age < 11')
mean_1st_range,mean_2nd_range
```

[38]:
```
(0.1523242721569889, 0.1331429950136722)
```

[49]:
```
#create data
y = [age_range_mean(1, 'Age < 11')]
for i in range(2, 13):
```

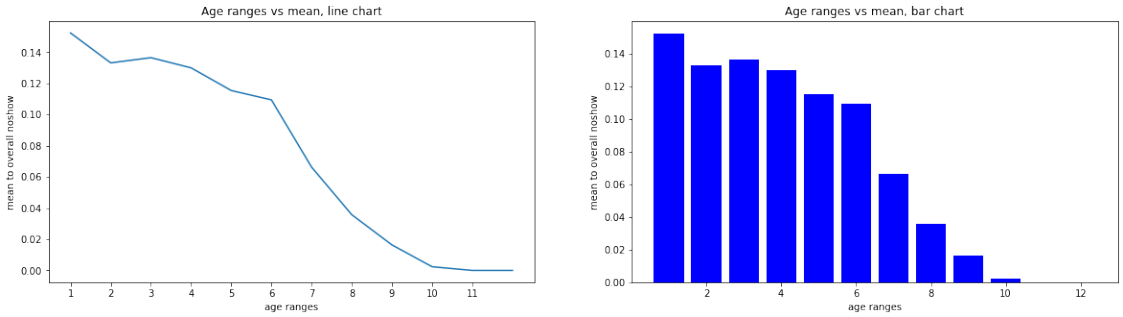
```
    range_mean = age_range_mean(i, 'Age < {}'.format(i*10 + 1), 'Age < {}'.
 ↪format((i-1)*10 +1))
    y.append(range_mean)
x = list(range(1,13))
```

[51]:
```
#plot the data
plt.figure(figsize=[20,5])

#1st plot
plt.subplot(1, 2, 1)
plt.plot(x, y)
plt.xticks(np.arange(1,12))
plt.xlabel('age ranges')
plt.ylabel('mean to overall noshow')
plt.title('Age ranges vs mean, line chart')

#2nd plot
plt.subplot(1, 2, 2)
plt.bar(x, y, color = 'blue')
plt.xlabel('age ranges')
plt.ylabel('mean to overall noshow')
plt.title('Age ranges vs mean, bar chart')
```

[51]:
```
Text(0.5, 1.0, 'Age ranges vs mean, bar chart')
```



From the above analysis and charts, our conclusions are: from the line chart as patient age gets older his chances to not show to his appontment gets less untill it approachs zero = 0 , and from the bar chart it's skewed to the left patients with small ages (young patients) have high chances to not show to their appointment (patient's `Age` < 11 have chance $= 15$ approximatly) and this value starts to get lower with age getting higher (untill we reach highest ages , `Age` > 90, the chances of not show reach zero).

1.1.4 Question 2 Who didn't show more to their appointment patients with hipertension, diabetes, alcoholism, or handcap?

[52]:
```
# create dataframe for people who show and who did not show
show = df['NoShow'] == 'No'
noshow = df['NoShow'] == 'Yes'
```

evalute value count for each variable (hipertension, diabetes, alcoholism, and handcap)

[53]:
```
#hipertension
df.Hipertension[show].value_counts()
df.Hipertension[noshow].value_counts()
```

[53]:
```
0    18547
1     3772
Name: Hipertension, dtype: int64
```

[54]:
```
#diabetes
df.Diabetes[show].value_counts()
df.Diabetes[noshow].value_counts()
```

[54]:
```
0    20889
1     1430
Name: Diabetes, dtype: int64
```

[55]:
```
#alcoholism
df.Alcoholism[show].value_counts()
df.Alcoholism[noshow].value_counts()
```

[55]:
```
0    21642
1      677
Name: Alcoholism, dtype: int64
```

[56]:
```
#handcap
df.Handcap[show].value_counts()
df.Handcap[noshow].value_counts()
```

[56]:
```
0    21912
1      407
Name: Handcap, dtype: int64
```

We can see that in every kind the number of shows is more than the number of noshow, and to compare noshow in the four with each other we'll use plot.

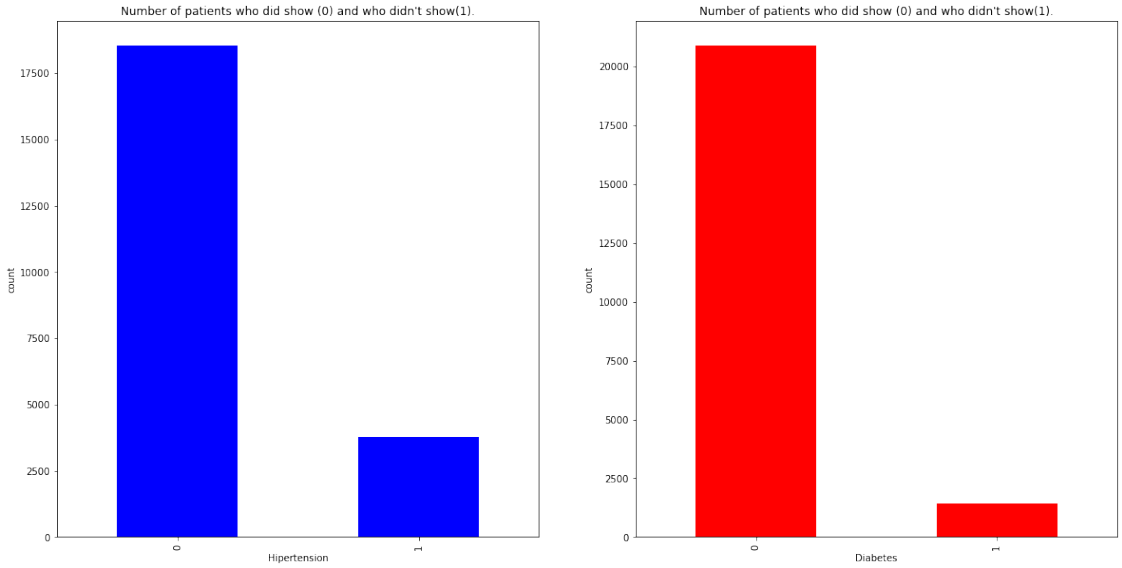
[60]:
```
plt.figure(figsize=[20,10])

plt.subplot(1, 2, 1)
```

```
df.Hipertension[noshow].value_counts().plot(kind = 'bar', color = 'blue', label␣
 ↪= 'Hipertension')
plt.xlabel('Hipertension')
plt.ylabel('count')
plt.title("Number of patients who did show (0) and who didn't show(1).")

plt.subplot(1, 2, 2)
df.Diabetes[noshow].value_counts().plot(kind = 'bar', color = 'red', label =␣
 ↪'Diabetes')
plt.xlabel('Diabetes')
plt.ylabel('count')
plt.title("Number of patients who did show (0) and who didn't show(1).")
```

[60]: Text(0.5, 1.0, "Number of patients who did show (0) and who didn't show(1).")



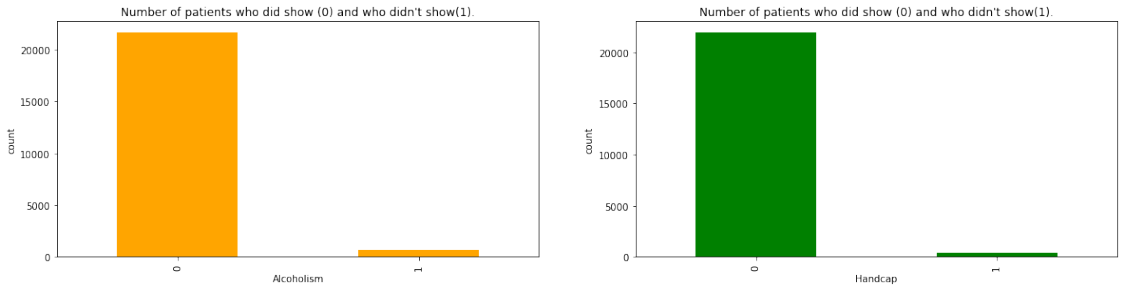
[61]:
```
plt.figure(figsize=[20,10])

plt.subplot(2, 2, 1)
df.Alcoholism[noshow].value_counts().plot(kind = 'bar', color = 'orange', label␣
 ↪= 'Alcoholism')
plt.xlabel('Alcoholism')
plt.ylabel('count')
plt.title("Number of patients who did show (0) and who didn't show(1).")

plt.subplot(2, 2, 2)
df.Handcap[noshow].value_counts().plot(kind = 'bar', color = 'green', label =␣
 ↪'Handcap')
plt.xlabel('Handcap')
```

```
plt.ylabel('count')
plt.title("Number of patients who did show (0) and who didn't show(1).")
```

[61]:
```
Text(0.5, 1.0, "Number of patients who did show (0) and who didn't show(1).")
```



pateints with hipertension tend to not show more than the rest, unlike patients with handcap who tend less to not show.

[ ]:
```
```

Calculating the mean for each type to overall noshow

[62]:
```
#Sum of all noshow corresponding to disease type
noshow_sum = df.query('Hipertension == 1')['NoShow'].value_counts().Yes + df.
 ↪query('Diabetes == 1')['NoShow'].value_counts().Yes + df.query('Alcoholism␣
 ↪== 1')['NoShow'].value_counts().Yes + df.query('Handcap == 1')['NoShow'].
 ↪value_counts().Yes
```

[63]:
```
#mean of Hipertension
mean_hipertension = df.query('Hipertension == 1')['NoShow'].value_counts().Yes /
 ↪ noshow_sum

#mean of Diabetes
mean_diabetes = df.query('Diabetes == 1')['NoShow'].value_counts().Yes /␣
 ↪noshow_sum

#mean of Alcoholism
mean_alcoholism = df.query('Alcoholism == 1')['NoShow'].value_counts().Yes /␣
 ↪noshow_sum

#mean of Handcap
mean_handcap = df.query('Handcap == 1')['NoShow'].value_counts().Yes /␣
 ↪noshow_sum
```

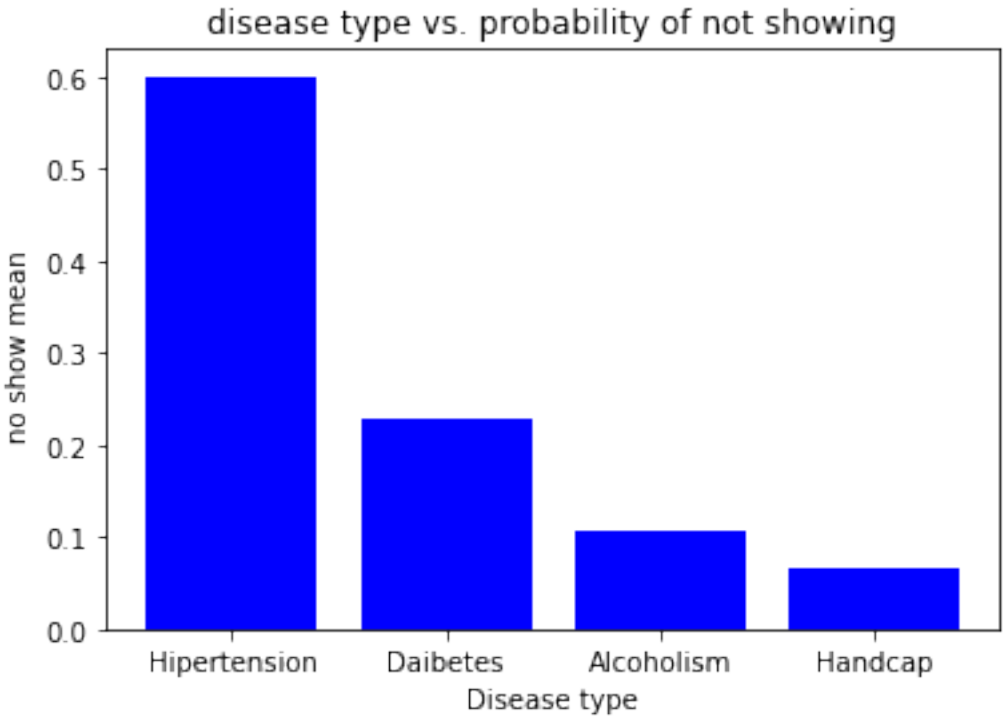
[64]:
```
mean_alcoholism, mean_diabetes, mean_handcap, mean_hipertension
```

[64]:
```
(0.10769965001590837,
 0.22748965956092904,
 0.06474705695195673,
 0.6000636334712058)
```

According to the mean, mean of `Hipertension` = 0.60 is the highest and mean of `Handcap` = 0.06 is the lowest and this is in confirm to the conclusion above.

[67]:
```
#plot the mean of the 4 types
plt.bar(x = 'Hipertension', height = mean_hipertension, color = 'blue')
plt.bar(x = 'Daibetes', height = mean_diabetes, color = 'blue')
plt.bar(x = 'Alcoholism', height = mean_alcoholism, color = 'blue')
plt.bar(x = 'Handcap', height = mean_handcap, color = 'blue')
plt.xlabel('Disease type')
plt.ylabel('no show mean ')
plt.title("disease type vs. probability of not showing")

plt.show()
```



patients with hypertension have higher probability(= 0.60) of not showing to their appointment, patients with diabetes comes next in line (probability = 0.22) and then patients wih Alcoholism (probability = 0.10) and at last patients with Handicap have probability of not showing = 0.06.

1.1.5 Question 3 Does having a scholarship have an effect if a patient shows or doesn't show to his appointment?

[68]:
```
#number of patients with scholarship and number of patients with no scholarship
df.Scholarship.value_counts()
```

[68]:
```
0    99666
1    10861
Name: Scholarship, dtype: int64
```

[69]:
```
#mean of patients with scholarship and those who doesn'nt have it
mean_scholarship = df.Scholarship.value_counts()[1]/ df.Scholarship.count()
mean_noscholarship = df.Scholarship.value_counts()[0]/ df.Scholarship.count()
mean_scholarship, mean_noscholarship
```

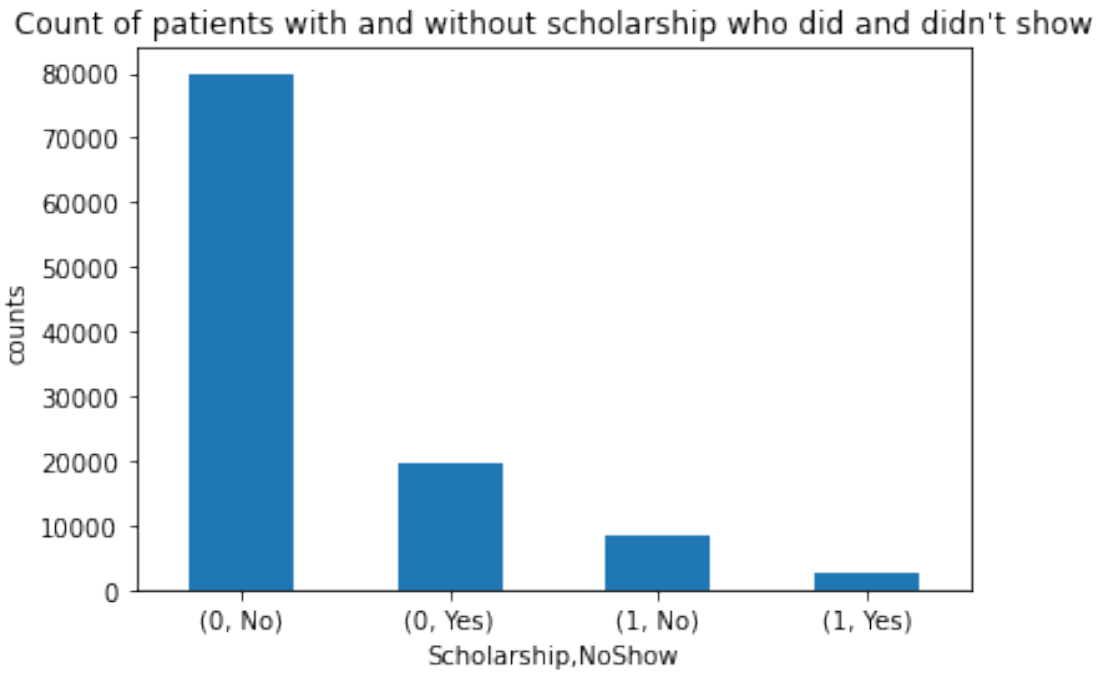
[69]:
```
(0.09826558216544373, 0.9017344178345562)
```

Number of patients with scholarship are less than these with no scholarship.

[70]:
```
df.groupby('Scholarship')['NoShow'].value_counts()
```

[70]:
```
Scholarship  NoShow
0            No        79925
             Yes       19741
1            No         8283
             Yes        2578
Name: NoShow, dtype: int64
```

[80]:
```
#plot the association of scholarship with No-show
df.groupby('Scholarship')['NoShow'].value_counts().plot(kind = 'bar');
plt.ylabel('counts')
plt.xticks(rotation = 0)
plt.title("Count of patients with and without scholarship who did and didn't␣
 ↪show");
```



This plot shows that patients with no scholarship tends more not to show, but we should use statics. Here we have to calculate it manually since no numeric values are used.

[72]:
```
noscholar = df.Scholarship == 0
scholar = df.Scholarship == 1
```

[73]:
```
#mean of patients with no scholarship and didn't show
mean_noSchol_noShow = ((df[noshow][noscholar].count())/(df.Scholarship[noshow].
 ↪count())).NoShow
mean_noSchol_noShow
```

```
/home/razan/anaconda3/envs/DAND/lib/python3.7/site-
packages/ipykernel_launcher.py:2: UserWarning: Boolean Series key will be
reindexed to match DataFrame index.
```

[73]:
```
0.8844930328419732
```

[74]:
```
#mean of patients with scholarship and didn't show
mean_Schol_noShow = ((df[noshow][scholar].count())/(df.Scholarship[noshow].
 ↪count())).NoShow
mean_Schol_noShow
```

```
/home/razan/anaconda3/envs/DAND/lib/python3.7/site-
packages/ipykernel_launcher.py:2: UserWarning: Boolean Series key will be
```

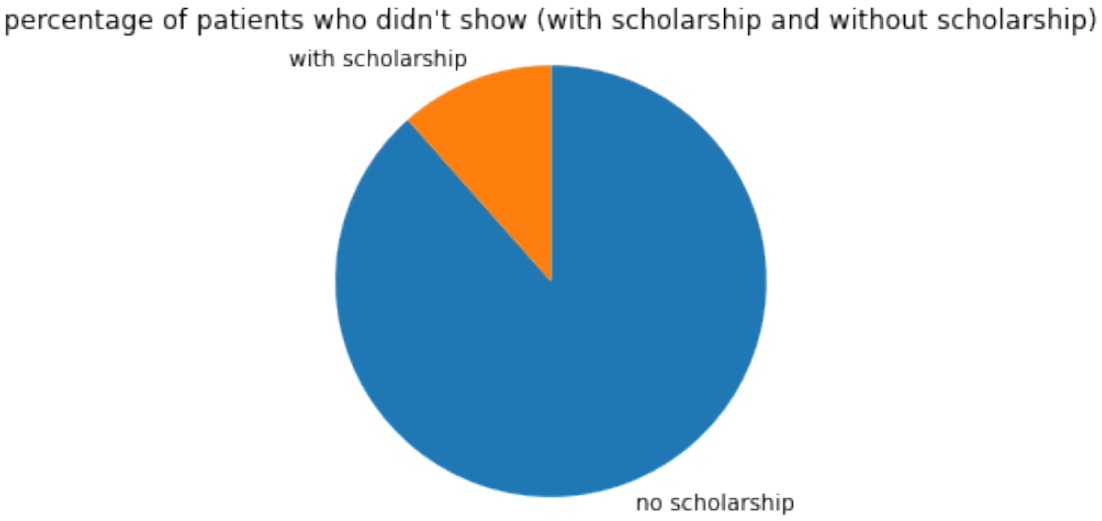
```
reindexed to match DataFrame index.
```

[74]:
```
0.11550696715802679
```

Statics confirm that patients with no Scholarship show more than patients with Scholarship.

[79]:
```
plt.pie([mean_noSchol_noShow, mean_Schol_noShow], labels = ['no␣
↪scholarship','with scholarship'], startangle = 90, counterclock = False);
plt.axis('square')
plt.title("percentage of patients who didn't show (with scholarship and without␣
↪scholarship)")
```

[79]:
```
Text(0.5, 1.0, "percentage of patients who didn't show (with scholarship and
without scholarship)")
```



Patients with no scholarship have higher probability (= 0.884) of not showing to their appointment than patients with scholarship (probability = 0.116).

Conclusions

1.1.6 Conclusions to First Quastion

Patients who age < 40 tends to not show to their appointments more than patients who age >= 40 with percentage of (0.601, 0.399) respectively. This is beacuas older people can fall to illness easier than young people and older patients take care of them selves more than young patients.

And if we looked more in deepth (more detailed age range), we can conclude a inverse proportionality between patient's `Age` and and not showing to his appointment "`NoShow`".

1.1.7 Conclusions to Second Quastion

Patients with Hypertension tend to not show to their appointment more than the rest with percentage of 0.600, while patients with handcap are the lest to not show to their appointment with parcentage of 0.065. This can be explained in response that handcap patients need more medical care than the rest, while patients with hypertension mantain their health by controlling their diet with healthy food and same as for patients with diabetes.

1.1.8 Conclusions to Third Quastion

Studying the association between Scholarship and NoShow varibales. Patients with no scholarship have higher probability (= 0.884) of not showing to their appointment than patients with scholarship (probability = 0.116).

It can be explained by two reseaons: 1st reason, usually patients without scholarship can't always medical bells, so their probabilty of not showing is higher than those who have scholarship. 2nd reason, number of patients who don't have scholarship are far more than patients who have it, so the probability of patient who doesn't have scholarship is higher.

[ ]: