Project: Investigate a Dataset (No-Show Appointments)

Table of Contents

Introduction

In this report we will investigate 110.527 medical appointments & its 14 associated variables (characteristics).

The following is the data description that i found on Kaggle Website.

01 - PatientId: Identification of a patient.

02 - AppointmentID: Identification of each appointment.

03 - Gender: Male or Female.

04 - Schedule day : the day the patient set up an appointment day in the future.

05 - Appointment day: the day the patient was expected to show up.

06 - Age: How old is the patient.

07 - Neighbourhood: Where the appointment takes place.

08 - Scholarship: True of False.

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.

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
from scipy.interpolate import interp1d

plt.style.use('seaborn')
pd.set_option('display.max_columns',200)
pd.set_option('display.max_rows',200)
```

Data Wrangling

In [ ]:

```
df=pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
df.sample(5)
```

Out[ ]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood
30321	9.658250e+12	5733006	F	2016-05-24T13:09:46Z	2016-05-25T00:00:00Z	48	RESISTÊNCIA
1657	9.529583e+10	5560893	M	2016-04-08T09:45:40Z	2016-04-29T00:00:00Z	2	JARDIM DA PENHA
100340	2.667720e+13	5770450	F	2016-06-03T10:19:02Z	2016-06-06T00:00:00Z	26	SÃO PEDRO
15227	7.544370e+13	5739261	F	2016-05-25T13:10:14Z	2016-05-30T00:00:00Z	29	JESUS DE NAZARETH
67583	1.496580e+13	5650166	F	2016-05-03T07:30:28Z	2016-05-03T00:00:00Z	5	SANTOS REIS

In [ ]:

```
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
```

So we have around 110,527 entries , none of them has null values which is good.
However, is there any duplicates or unqiues among patientID and AppointmentID? possibaly we can use one of them as an index to our dataframe?

For the sake of convenience , i'll lower the cases for all column names and use underscores to seperate between two words.

In [ ]:

```
df.rename(str.lower, axis='columns' ,inplace=True)
```

```
df.rename({'no-show':'no_show'}, axis='columns' ,inplace=True)
df.head()
```

Out[ ]:

	patientid	appointmentid	gender	scheduledday	appointmentday	age	neighbourhood	scholars
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	
1	5.589980e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	
2	4.262960e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA	
3	8.679510e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI	
4	8.841190e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	

In [ ]:

```
print('patient id duplicates:',df.patientid.duplicated().sum())
print('appointment id duplicates:',df.appointmentid.duplicated().sum())
```

```
patient id duplicates: 48783
appointment id duplicates: 0
```

Looks like Appointment ID has no duplicates so we can use it as an index.

Patient ID duplicating make sense , it means that the same patient had several appointments.

In [ ]:

```
df.set_index('appointmentid' , inplace=True)
df.sample(1)
```

Out[ ]:

	patientid	gender	scheduledday	appointmentday	age	neighbourhood	scholarship
appointmentid							
5666701	5.714350e+13	M	2016-05-06T06:55:01Z	2016-05-10T00:00:00Z	52	GRANDE VITÓRIA	0

In [ ]:

```
print('Number of Patients with unique ID:',len(df.patientid.unique()))
print('Number of Appointments:',df.patientid.value_counts().sum())
```

```
Number of Patients with unique ID: 61744
Number of Appointments: 110527
```

Data Cleaning

Do duplicate entries exist among our data frame?

In [ ]:

```
print('Number of entries before removing dulicates:',df.shape[0])
print('Duplicated entries:',df.duplicated().sum())
df.drop_duplicates(inplace=True)
print('Number of entries after removing dulicates:',df.shape[0])
```

```
Number of entries before removing dulicates: 110527
Duplicated entries: 618
Number of entries after removing dulicates: 109909
```

Now lets check the unique values for each column to verify data integrity, this helps me to avoid any future hidden surprises.

```
In [ ]:
for column in df.columns.drop(['patientid','scheduledday','appointmentday']):
    print(f'{column}:{df[column].unique()}\n')

print(f'Value Counts:{df[df.age==-1].age.value_counts()}')
```

```
gender:['F' 'M']

age:[ 62  56   8  76  23  39  21  19  30  29  22  28  54  15  50  40  46   4
  13  65  45  51  32  12  61  38  79  18  63  64  85  59  55  71  49  78
  31  58  27   6   2  11   7   0   3   1  69  68  60  67  36  10  35  20
  26  34  33  16  42   5  47  17  41  44  37  24  66  77  81  70  53  75
  73  52  74  43  89  57  14   9  48  83  72  25  80  87  88  84  82  90
  94  86  91  98  92  96  93  95  97 102 115 100  99  -1]

neighbourhood:['JARDIM DA PENHA' 'MATA DA PRAIA' 'PONTAL DE CAMBURI' 'REPÚBLICA'
 'GOIABEIRAS' 'ANDORINHAS' 'CONQUISTA' 'NOVA PALESTINA' 'DA PENHA'
 'TABUAZEIRO' 'BENTO FERREIRA' 'SÃO PEDRO' 'SANTA MARTHA' 'SÃO CRISTÓVÃO'
 'MARUÍPE' 'GRANDE VITÓRIA' 'SÃO BENEDITO' 'ILHA DAS CAIEIRAS'
 'SANTO ANDRÉ' 'SOLON BORGES' 'BONFIM' 'JARDIM CAMBURI' 'MARIA ORTIZ'
 'JABOUR' 'ANTÔNIO HONÓRIO' 'RESISTÊNCIA' 'ILHA DE SANTA MARIA'
 'JUCUTUQUARA' 'MONTE BELO' 'MÁRIO CYPRESTE' 'SANTO ANTÔNIO' 'BELA VISTA'
 'PRAIA DO SUÁ' 'SANTA HELENA' 'ITARARÉ' 'INHANGUETÁ' 'UNIVERSITÁRIO'
 'SÃO JOSÉ' 'REDENÇÃO' 'SANTA CLARA' 'CENTRO' 'PARQUE MOSCOSO'
 'DO MOSCOSO' 'SANTOS DUMONT' 'CARATOÍRA' 'ARIOVALDO FAVALESSA'
 'ILHA DO FRADE' 'GURIGICA' 'JOANA D´ARC' 'CONSOLAÇÃO' 'PRAIA DO CANTO'
 'BOA VISTA' 'MORADA DE CAMBURI' 'SANTA LUÍZA' 'SANTA LÚCIA'
 'BARRO VERMELHO' 'ESTRELINHA' 'FORTE SÃO JOÃO' 'FONTE GRANDE'
 'ENSEADA DO SUÁ' 'SANTOS REIS' 'PIEDADE' 'JESUS DE NAZARETH'
 'SANTA TEREZA' 'CRUZAMENTO' 'ILHA DO PRÍNCIPE' 'ROMÃO' 'COMDUSA'
 'SANTA CECÍLIA' 'VILA RUBIM' 'DE LOURDES' 'DO QUADRO' 'DO CABRAL' 'HORTO'
 'SEGURANÇA DO LAR' 'ILHA DO BOI' 'FRADINHOS' 'NAZARETH' 'AEROPORTO'
 'ILHAS OCEÂNICAS DE TRINDADE' 'PARQUE INDUSTRIAL']

scholarship:[0 1]

hipertension:[1 0]

diabetes:[0 1]

alcoholism:[0 1]

handcap:[0 1 2 3 4]

sms_received:[0 1]

no_show:['No' 'Yes']

Value Counts:-1    1
Name: age, dtype: int64
```

I noticed multiple issues with the Unique values and they are:

A) There is one unique value in age which is -1 , that doesnt make any sense.

B) Why does handcap column has unique values from 0 to 4? , my first guess was it would be a binary stating if patient is handicaped or not as in 1 and 0.

C) sms_recieved column has 0 and 1 values only indicating that its binary , however on the data description, it states that it can varry from 0 to 5 indicating number of sms sent , was it swapped out with handcap column?

A) My first guess is that the -1 might had been a misentry , a 1 year old kid is less likely to have hipertension , diabetes & alchololism issues , the following cells show clearly the sum of 1 year old kids and how many of them suffer from certain issues , my fix would be is changing the -1 into 1

In [ ]:

```
print("sum of people older than 1 year")
print(df[df.age!=1].iloc[:,7:11].sum())
print("sum of people that are 1 year old")
print(df[df.age==1].iloc[:,7:11].sum())
```

```
sum of people older than 1 year
hipertension    21678
diabetes         7892
alcoholism       3344
handcap          2431
dtype: int64
sum of people that are 1 year old
hipertension    0
diabetes        1
alcoholism      0
handcap         1
dtype: int64
```

In [ ]:

```
df.loc[df.age == -1,'age'] = 1
df[df.age == -1]
```

Out[ ]:

	patientid	gender	scheduledday	appointmentday	age	neighbourhood	scholarship	hi
appointmentid								

B) After doing a little research on kaggle , i found this message from the dataset creator stating that the numbers in handcap column refers to the amount of disabilities that the patient is suffering from. so nothing to be changed.

In [ ]:

```
print('number of unique values')
df.handcap.value_counts()
```

```
number of unique values
```

Out[ ]:

```
0    107690
1      2023
2       182
3        11
4         3
Name: handcap, dtype: int64
```

C) Prior to the previous mark down , it clearly states that 0 means No sms recieved and 1 means message sent , so i will change nothing.

In [ ]:
```
df.sms_received.value_counts()
print('Number ofduplicates:',df.duplicated().sum())
```

```
Number ofduplicates: 0
```

Setting DataFrame dtypes

Changing Date to datetime object with format : year - month - day

In [ ]:
```
df.appointmentday=pd.to_datetime(df.appointmentday).dt.date
df.scheduledday=pd.to_datetime(df.scheduledday).dt.date
df.head(1)
print('Number ofduplicates:',df.duplicated().sum())
```

```
Number ofduplicates: 3604
```

Setting up a new column (waiting_days) to describe how many days between:

Schedule day : the day the patient set up an appointment day in the future.
Appointment day: the day the patient was expected to show up.

In [ ]:
```
df['waiting_days']=df.appointmentday - df.scheduledday
df['waiting_days']=df.waiting_days.dt.days
df.waiting_days.value_counts().sort_index().head()
```

Out[ ]:
```
-6        1
-1        4
 0    38495
 1     5162
 2     6698
Name: waiting_days, dtype: int64
```

Looks like we have 5 invalid entries, one (-6 days) and four (-1 days) , I'm not sure why they exist , but I decided to remove them.

In [ ]:
```
df=df.drop(df[df.waiting_days< 0].index)
df.waiting_days.value_counts().sort_index().head()
```

Out[ ]:
```
0    38495
1     5162
2     6698
3     2711
4     5269
Name: waiting_days, dtype: int64
```

Waiting Days looks sorted now

Exploratory Data Analysis

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

Out[ ]:

	patientid	age	scholarship	hipertension	diabetes	alcoholism	1099(
count	1.099040e+05	109904.000000	109904.000000	109904.000000	109904.000000	109904.000000	1099(
mean	1.474549e+14	37.086657	0.098286	0.197245	0.071817	0.030427	
std	2.560406e+14	23.121320	0.297702	0.397921	0.258186	0.171759	
min	3.920000e+04	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	4.172980e+12	18.000000	0.000000	0.000000	0.000000	0.000000	
50%	3.172725e+13	37.000000	0.000000	0.000000	0.000000	0.000000	
75%	9.439392e+13	55.000000	0.000000	0.000000	0.000000	0.000000	
max	9.999820e+14	115.000000	1.000000	1.000000	1.000000	1.000000	

In [ ]:

In [ ]:

In [ ]:

```
cribe()
```

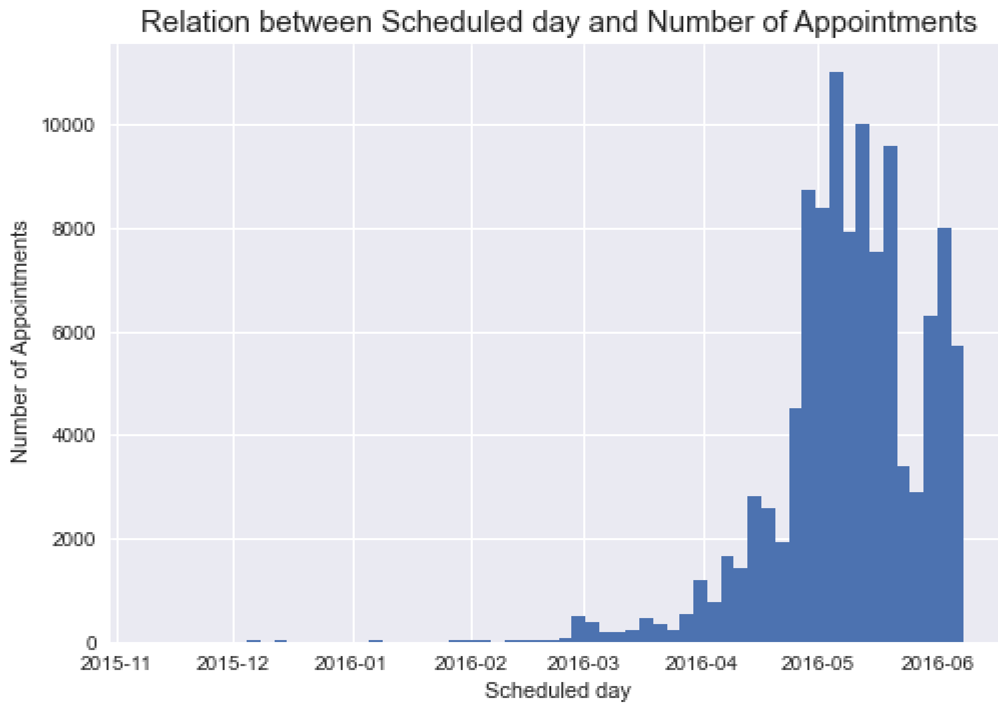
From the data above we can see that the average age is 37 , atleats 75% of the data population doesn't have hipertension,diabetes,scolarship program,not alcoholic and not hanidcaped, atleast 50% of the data didn't receive sms, the average waiting days for an appointment is 10 days, atleast 50% of the data population has 4 days waiting for an appointment.

In [ ]:

```
df.scheduledday.hist(bins=60)
plt.xlabel('Scheduled day')
plt.ylabel(' Number of Appointments')
plt.title('Relation between Scheduled day and Number of Appointments' , fontsize=15);
```

Out[ ]:

```
Text(0.5, 1.0, 'Relation between Scheduled day and Number of Appointments')
```



In [ ]:

```
time_outliers=df[df.scheduledday<datetime.date(year=2016,month=3,day=1)]
len(time_outliers)
```

Out[ ]:

```
402
```

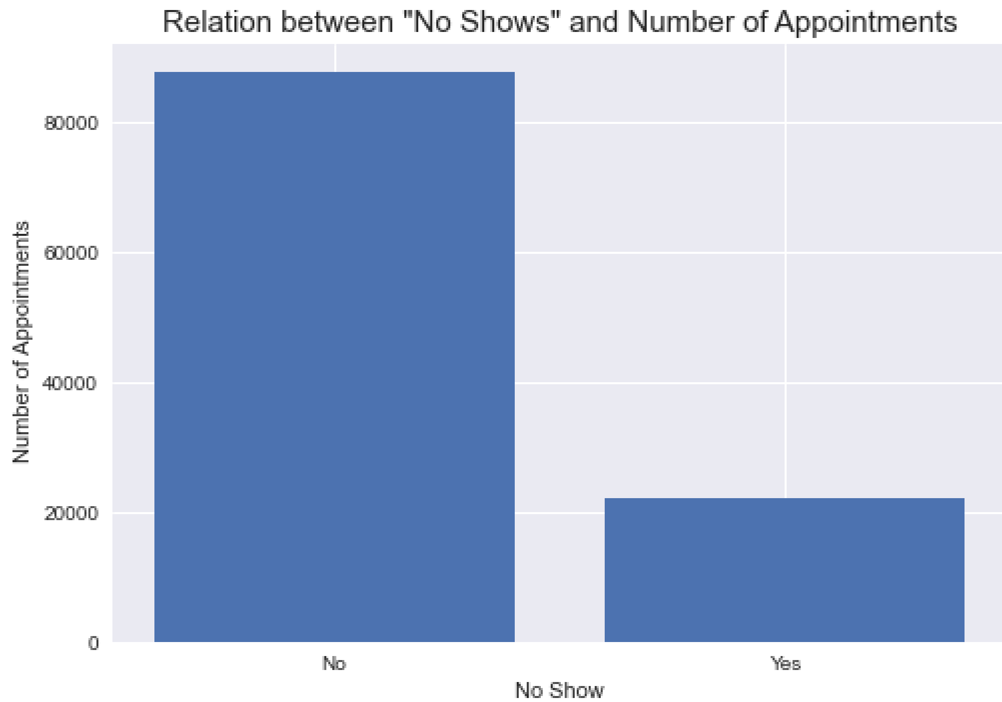
Looks like the scheduled day is left skewed , why most of the appointments suddenly happened after march 2016?

In [ ]:

```
data_set_1=df.no_show.value_counts()
plt.bar(data_set_1.index,data_set_1)
plt.xlabel('No Show')
plt.ylabel(' Number of Appointments')
plt.title('Relation between "No Shows" and Number of Appointments' , fontsize=15)
data_set_1
```

Out[ ]:

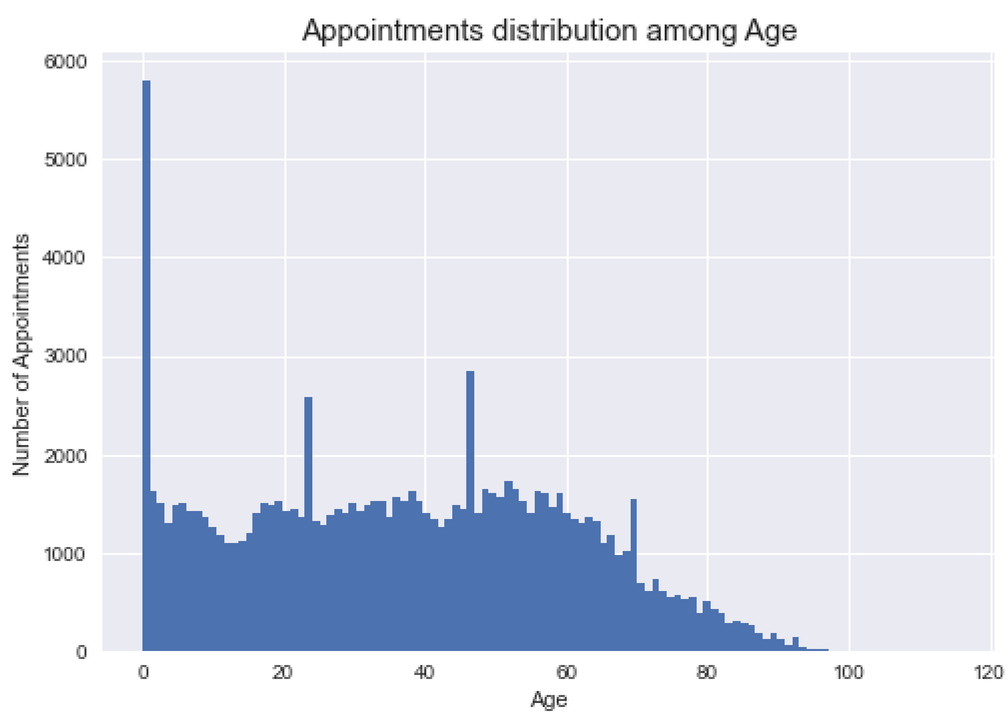
```
No      87804
Yes     22100
Name: no_show, dtype: int64
```



From the chart above, we can see that 87,804 of the appointments are 'show up' while 22,100 are 'no shows'

In [ ]:

```
df.age.hist(bins=110)
plt.xlabel('Age')
plt.ylabel('Number of Appointments')
plt.title('Appointments distribution among Age' , fontsize=15);
```



In [ ]:

```
plt.boxplot(df.age);
plt.ylabel('Age')
plt.title('Appointments distribution among Age' , fontsize=15);
```

The boxplot above shows that the age ranges from 0 up to 98 years with median of 37 years old. outliers beyond ~100 years.\ where kids with less than a year has highest appointments.

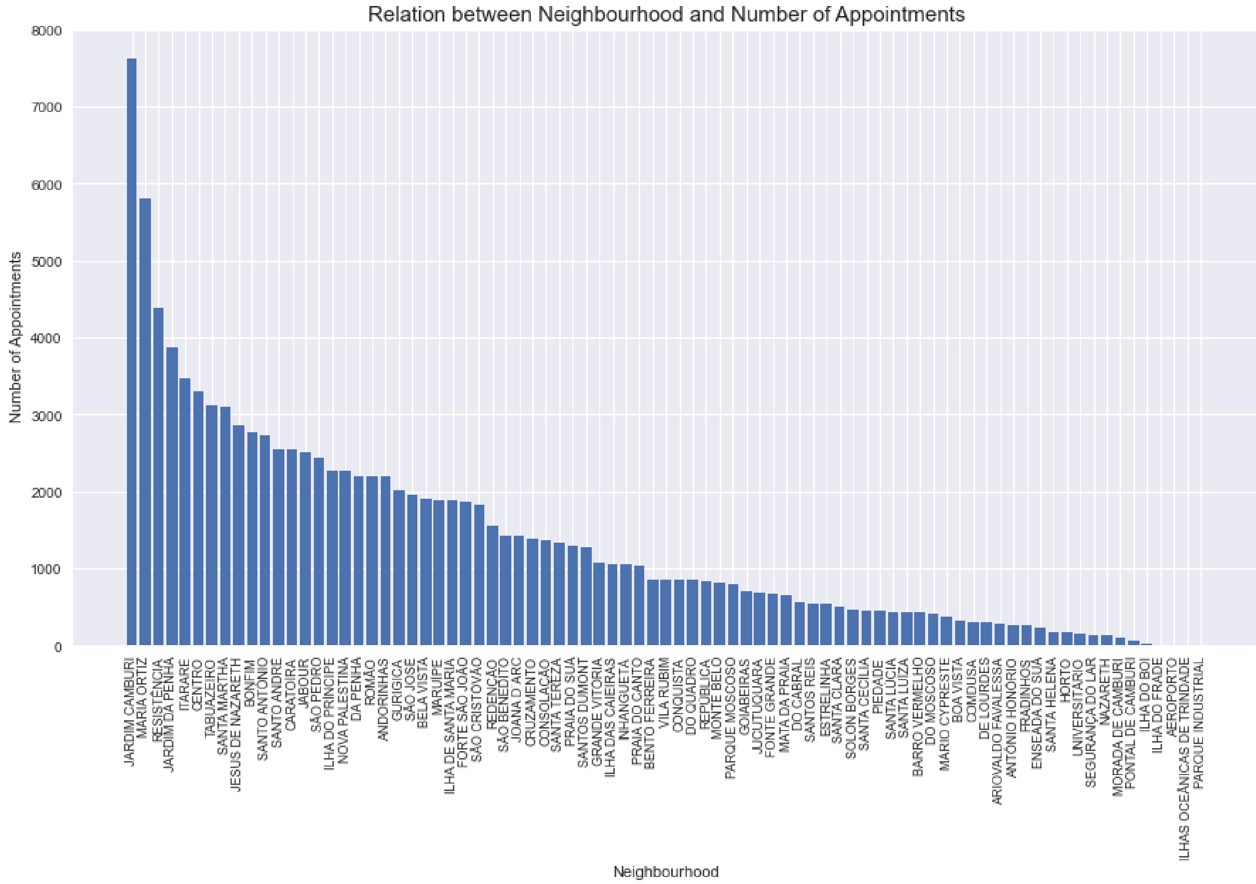
In [ ]:

```
plt.figure(figsize=(15,8))
data_set_2=df.neighbourhood.value_counts()
plt.bar(data_set_2.index,data_set_2)
plt.xlabel('Neighbourhood')
plt.ylabel(' Number of Appointments',)
plt.xticks(rotation=90,fontsize=9)
plt.title('Relation between Neighbourhood and Number of Appointments' , fontsize=15)
data_set_2
```

Out[ ]:

```
JARDIM CAMBURI          7621
MARIA ORTIZ             5804
RESISTÊNCIA             4386
JARDIM DA PENHA         3873
ITARARÉ                 3470
CENTRO                  3310
TABUAZEIRO              3122
SANTA MARTHA            3103
JESUS DE NAZARETH       2852
BONFIM                  2761
SANTO ANTÔNIO           2731
SANTO ANDRÉ             2556
CARATOÍRA               2541
JABOUR                  2507
SÃO PEDRO               2432
ILHA DO PRÍNCIPE        2266
NOVA PALESTINA          2263
DA PENHA                2203
ROMÃO                   2197
ANDORINHAS              2194
GURIGICA                2014
SÃO JOSÉ                1963
BELA VISTA              1894
MARUÍPE                 1891
ILHA DE SANTA MARIA     1885
```

```
FORTE SÃO JOÃO                 1867
SÃO CRISTÓVÃO                  1831
REDENÇÃO                       1553
SÃO BENEDITO                   1433
JOANA D´ARC                    1420
CRUZAMENTO                     1387
CONSOLAÇÃO                     1368
SANTA TEREZA                   1327
PRAIA DO SUÁ                   1288
SANTOS DUMONT                  1274
GRANDE VITÓRIA                 1071
ILHA DAS CAIEIRAS              1060
INHANGUETÁ                     1057
PRAIA DO CANTO                 1032
BENTO FERREIRA                  854
VILA RUBIM                      851
CONQUISTA                       847
DO QUADRO                       846
REPÚBLICA                       835
MONTE BELO                      824
PARQUE MOSCOSO                  797
GOIABEIRAS                      699
JUCUTUQUARA                     694
FONTE GRANDE                    675
MATA DA PRAIA                   643
DO CABRAL                       558
SANTOS REIS                     542
ESTRELINHA                      538
SANTA CLARA                     501
SOLON BORGES                    469
SANTA CECÍLIA                   448
PIEDADE                         446
SANTA LÚCIA                     436
SANTA LUÍZA                     428
BARRO VERMELHO                  422
DO MOSCOSO                      411
MÁRIO CYPRESTE                  367
BOA VISTA                       312
COMDUSA                         303
DE LOURDES                      302
ARIOVALDO FAVALESSA             280
ANTÔNIO HONÓRIO                 271
FRADINHOS                       258
ENSEADA DO SUÁ                  235
SANTA HELENA                    178
HORTO                           175
UNIVERSITÁRIO                   152
SEGURANÇA DO LAR                144
NAZARETH                        135
MORADA DE CAMBURI                96
PONTAL DE CAMBURI                69
ILHA DO BOI                      35
ILHA DO FRADE                    10
AEROPORTO                         8
ILHAS OCEÂNICAS DE TRINDADE       2
PARQUE INDUSTRIAL                 1
Name: neighbourhood, dtype: int64
```



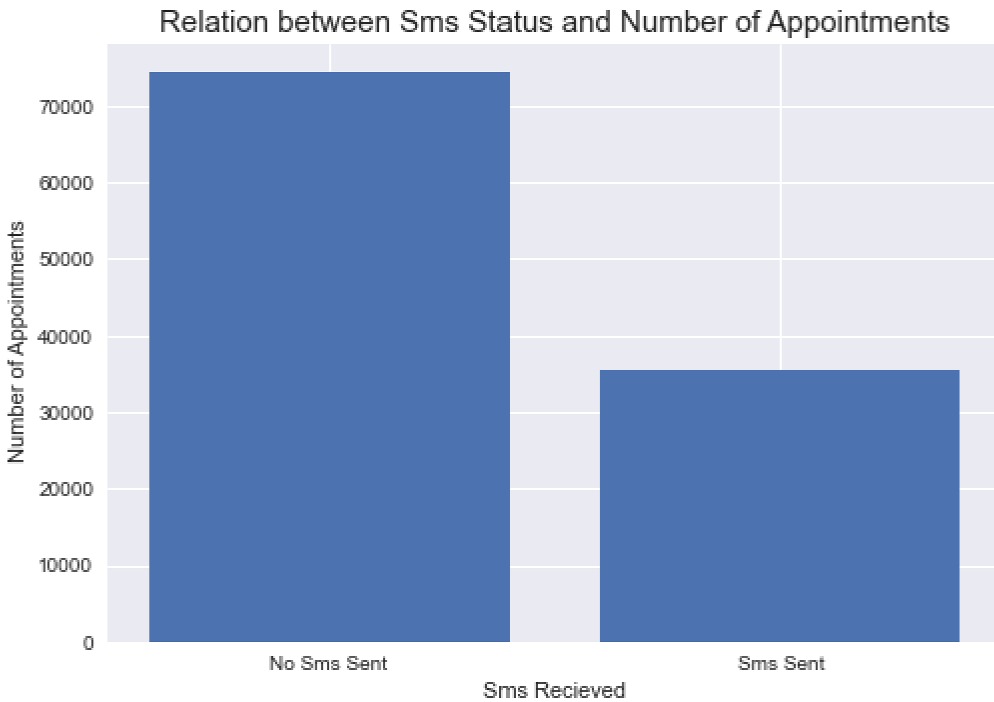
Distribution for appointments among various neighbourhoods, certain neighbourhoods have significantly higher appointments than others.

In [ ]:

```
daata_set_3=df.sms_received.value_counts()
plt.bar(daata_set_3.index,daata_set_3)
plt.xlabel('Sms Recieved')
plt.xticks(daata_set_3.index,['No Sms Sent','Sms Sent'])
plt.ylabel(' Number of Appointments')
plt.title('Relation between Sms Status and Number of Appointments' , fontsize=15)
daata_set_3
```

Out[ ]:

```
0    74422
1    35482
Name: sms_received, dtype: int64
```



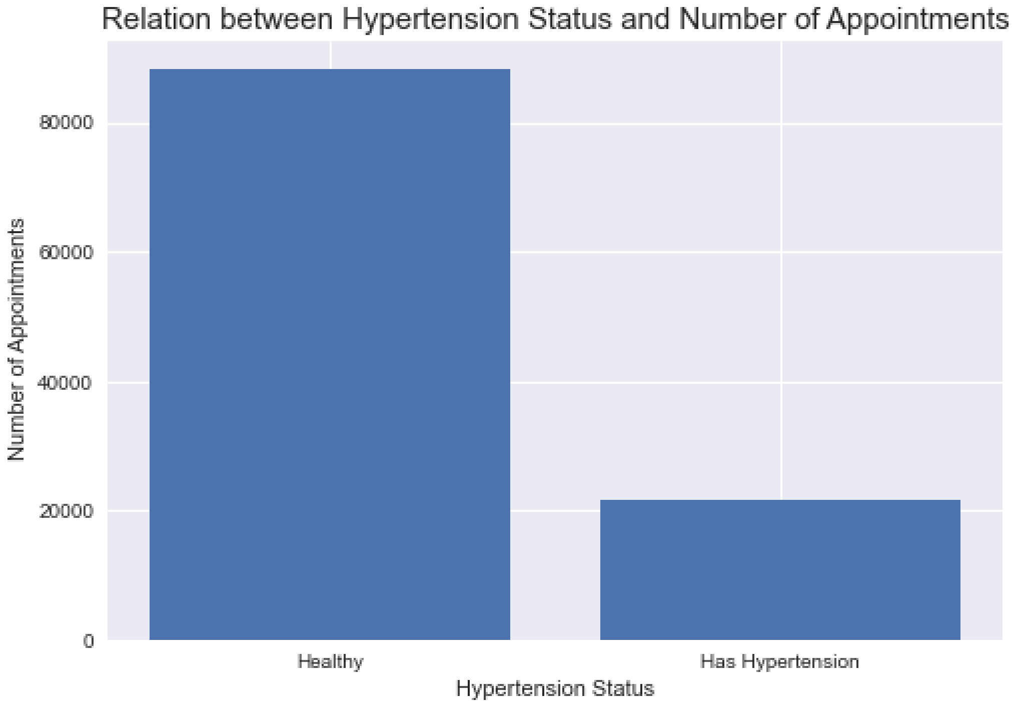
From the chart above, we can see that 74,422 Appointments didn't get notified with sms, Where 35,482 appointments were sent an sms.

In [ ]:

```
data_set_4=df.hipertension.value_counts()
plt.bar(data_set_4.index,data_set_4)
plt.xlabel('Hypertension Status')
plt.xticks(data_set_4.index,['Healthy','Has Hypertension'])
plt.ylabel(' Number of Appointments')
plt.title('Relation between Hypertension Status and Number of Appointments' , fontsize=
data_set_4
```

Out[ ]:

```
0    88226
1    21678
Name: hipertension, dtype: int64
```



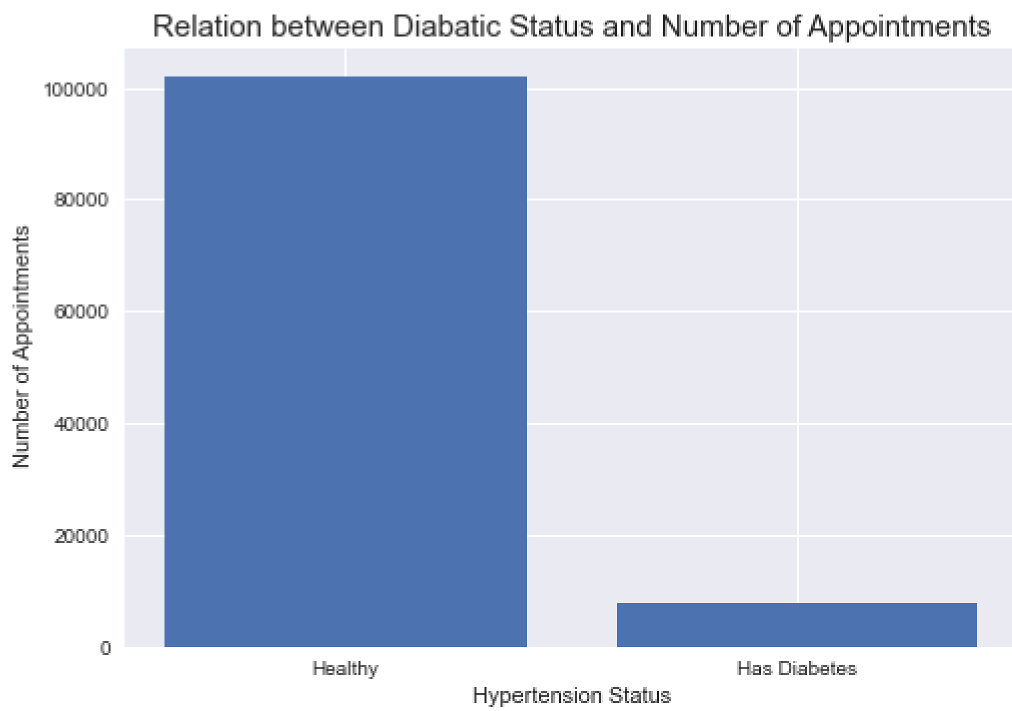
From the chart above we can see that 88,226 from the appointments had patients with no hypertension issues , where 21,678 of the appointments had patients suffering from hypertension issues.

In [ ]:

```
data_set_5=df.diabetes.value_counts()
plt.bar(data_set_5.index,data_set_5)
plt.xlabel('Hypertension Status')
plt.xticks(data_set_5.index,['Healthy','Has Diabetes'])
plt.ylabel(' Number of Appointments')
plt.title('Relation between Diabatic Status and Number of Appointments' , fontsize=15)
data_set_5
```

Out[ ]:

```
0    102011
1      7893
Name: diabetes, dtype: int64
```



From the chart above we can see that 102,011 from the appointments had patients with no diabatic issues , where 7,893 of the appointments had patients suffering from diabetes.

In [ ]:

```
def spline(data,ax,smoothness):
    """[summary]: This function creates a spline using a pandas series and selecting an
    Args:
        data (pandas.core.series.Series): pandas series containing the data used to cre
        ax : axe name to be assigned to , i.e ax1, ax2 ...etc
    """
    f = interp1d(data.index, data, kind='quadratic')
    x_new = np.linspace(data.index.min(), data.index.max(),smoothness)
    y_smooth=f(x_new)
    ax.plot(x_new,y_smooth,color='orange')
```

Research Question 1 (Is there a relation between the number of waiting days for an appointment and the 'no-show' rate?)

I will set up masks to filter my DataFrame so i can select the indexes of Appointments who showed /didn't show up.

showed_up returns pandas series holding indexes and the corosponding truth values, True if it finds no_show Column meeting the condtion in this case " no_show == 'No' " , while the didnt_show_up mask is vise versa.

In [ ]:

```
showed_up = df.no_show == 'No'
didnt_show_up= df.no_show == 'Yes'
```

Now based on these masks i will create another two pandas series to show me how many Appointments showed / Didn't show up for every range of waiting days like so:

In [ ]:

```
plot1=df.waiting_days[showed_up].value_counts().sort_index()
```

```
plot2=df.waiting_days[didnt_show_up].value_counts().sort_index()
print(plot1.head())
plot2.head()
```

Out[ ]:

```
0    36712
1     4063
2     5103
3     2076
4     4046
Name: waiting_days, dtype: int64
0    1783
1    1099
2    1595
3     635
4    1223
Name: waiting_days, dtype: int64
```

I will also Calculate the sum of waiting days for every day sample for both Appointments which showed up and those which didn't . this will be useful in order to calculate the percentage for each group of waiting days.

In [ ]:

```
total_day=df.waiting_days.value_counts().sort_index()
total_day.head()
```

Out[ ]:

```
0    38495
1     5162
2     6698
3     2711
4     5269
Name: waiting_days, dtype: int64
```

Now, we can actually see the percentage of Appointments who showed up to the ones which didn't.\ Note that the sum of any percantage for any specific number of waiting days for who showed up and who didn't must be equal to one.\ i.e: 0 days hold 0.953 in the 1st Series and 0.0463 in the 2nd Series , their sum is equal to 1.

In [ ]:

```
plot1_pst=plot1.divide(total_day,fill_value=0)
plot2_pst=plot2.divide(total_day,fill_value=0)
print(plot1_pst.head())
plot2_pst.head()
```

Out[ ]:

```
0    0.953682
1    0.787098
2    0.761869
3    0.765769
4    0.767888
Name: waiting_days, dtype: float64
0    0.046318
1    0.212902
2    0.238131
3    0.234231
4    0.232112
Name: waiting_days, dtype: float64
```

Now, I will create two sub plots ,the 1st show us the actual number of people who showed / didn't show up related to how large was their waiting days for their appointment. the last one show us the actual percentage for Appointments which showed up to the ones which didn't.

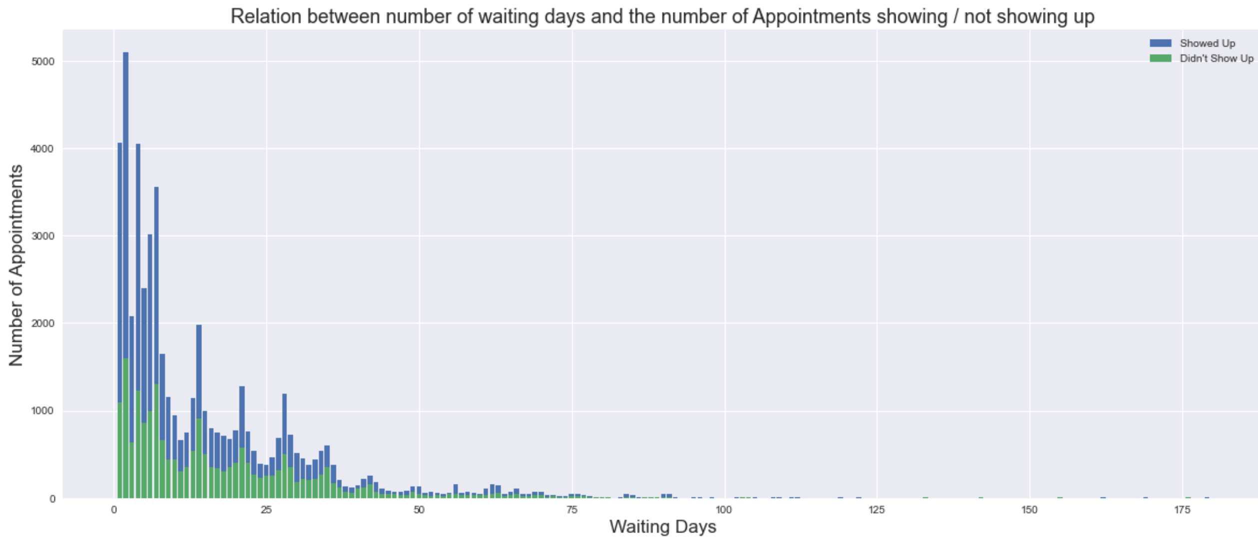
In [ ]:

```
fig1, (ax1, ax2) = plt.subplots(2,1,figsize=(20,16));
ax1.set_title('Relation between number of waiting days and the number of Appointments s
ax1.bar(plot1.index , plot1 ,label='Showed Up');
ax1.bar(plot2.index , plot2, label="Didn't Show Up");
ax1.set_xlabel('Waiting Days',fontsize=17);
ax1.set_ylabel('Number of Appointments',fontsize=17);
ax1.tick_params(labelrotation=0 , labelsize=15);
ax1.legend(loc=1);

ax2.bar(plot1_pst.index , plot1_pst, label='Showed Up');
ax2.bar(plot2_pst.index , plot2_pst, label="Didn't Show Up");
spline(plot2_pst,ax2,500)
ax2.set_xlabel('Waiting Days',fontsize=17);
ax2.set_ylabel('Percentage of Appointments showing / not showing up',fontsize=17);
ax2.tick_params(labelrotation=0,labelsize=15);
```

I will plot the 1st plot again but without "0 days" waiting time to further elaborate the relations.

In [ ]:

```
plt.figure(figsize=(20,8));
plt.title('Relation between number of waiting days and the number of Appointments showi
plt.bar(plot1.index[1:] , plot1[1:] ,label='Showed Up');
plt.bar(plot2.index[1:] , plot2[1:], label="Didn't Show Up");
plt.xlabel('Waiting Days',fontsize=17);
plt.ylabel('Number of Appointments',fontsize=17);
plt.legend(loc=1);
```



Q1 Observation & Conclusions:

> **First plot**: The 1st bar '0' indicated that about 35,000+ of the appointments (out of the 109904 entries) happened in the same day and they were showed up for, this might indicates medical urgency or more attention for appointment date since it is in the same day, whereas less than 2000 Appointments didn't show up , after that, the bars drop down to 5000 Appointments mark, where the Appointments with around 10 days of waiting shows significant 'show ups', after 10 days of waiting time , it looks like 'no-show' values become very close to the 'showed'.
>
> **Second plot**: The 1st bar clearley states that Around 95% of the appointments on the same day 'shows up', where 5 % dont ,moving on we have a jump in 'no-shows' from 5% to 21% for the appointments with 1 day waiting time , the percentage increase in a semi-linear fashion up until 31% 'no-shows'at 14 days wating time mark, after that, the percentage keeps on fluctuating 40% and 28% 'no-shows' up until 56 days waiting time mark , moving on the fluctuation is even wilder with upper limit 44% and lower limit 22% up until the 69 days waiting time mark.
>
>> I think that it is safe to say that we can " partially calculate" the percantage of someone not showing up for an appointment based on waiting days if they ask for it during a month period , why a month? , we can see that the 1st plot is skewed to the right , majority of the data samples exist during the 1st month and around 30% of the data is just in the 1st tick "0 days" , so if i would make decision based on a guess , it has to be during the 1st month , now by saying "partially calculate" i mean that there might be other factors influencing "no-shows", not just the waiting days.

Research Question 2 (Is there a relation between the age and the 'no-show' rate?)

In [ ]:

```
plot3=df.age[showed_up].value_counts()
```

```
plot4=df.age[didnt_show_up].value_counts()

total_age=plot3.add(plot4,fill_value=0).astype(int)

plot3_pst=plot3.divide(total_age,fill_value=0)
plot4_pst=plot4.divide(total_age,fill_value=0)

fig2, (ax1, ax2) = plt.subplots(2,1,figsize=(20,16));

ax1.set_title("Relation between age and the number of appointments showed / Didn't show
ax1.bar(plot3.index , plot3 ,label='Showed Up');
ax1.bar(plot4.index , plot4, label="Didn't Show Up");
ax1.set_ylabel('Number of appointments',fontsize=17);
ax1.tick_params(labelrotation=0 , labelsize=15);
ax1.legend(loc=1);
#ax1.set_xticks(plot1.index)

ax2.bar(plot3_pst.index , plot3_pst, label='Showed Up');
ax2.bar(plot4_pst.index , plot4_pst, label="Didn't Show Up");
spline(plot4_pst,ax2,200)
ax2.set_xlabel('Age',fontsize=17);
ax2.set_ylabel("Percentage of appointments showed / Didn't show up",fontsize=17);
ax2.tick_params(labelrotation=0,labelsize=15);
#ax2.set_xticks(plot1_pst.index)
```

In [ ]:

```
df.groupby('no_show')['age'].mean()
```

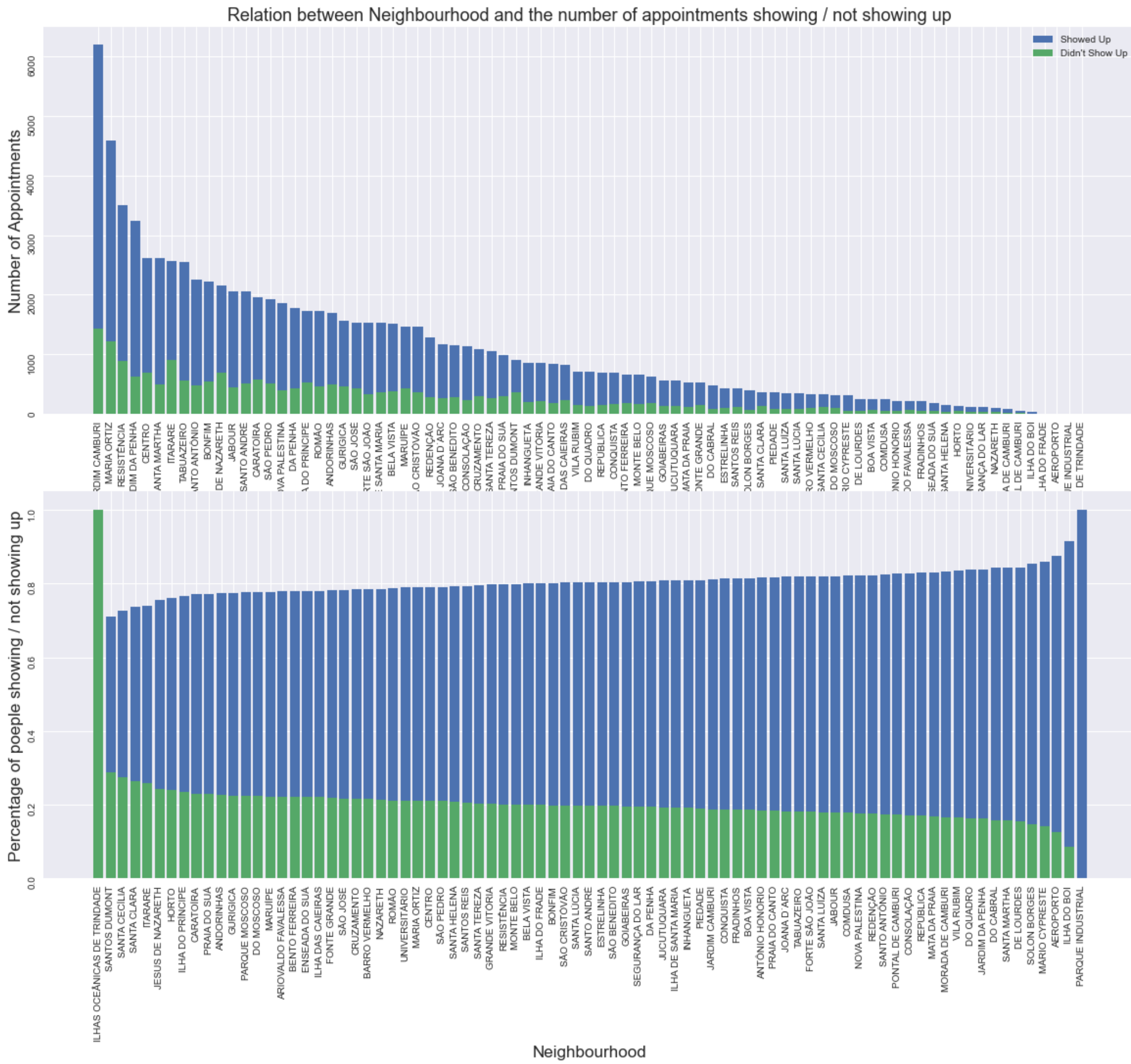
Out[ ]:
```
no_show
No     37.792697
Yes    34.281538
Name: age, dtype: float64
```

Q2 Observation and Conclusion

> **First plot**: looks like kids that are younger than 1 year old is the maximum count and its around 50% more than the other ages, counts decreases as age increases till around the 14~15 years old mark , then it keeps on increasings till 19~20 years old mark .moving on, the count of 'show up' appointments keeps on increasing till around 55~60 years old , after that less people appointments in every age category.in general, it looks like older people of average 38 years old seem to show up more than who don't with an avarge of 34 years old.
>
> **Second plot**: the'no-shows' peaks at 14~15 years old mark with a percentage little less than 30% and its keeps on gradually decreasing all the way till 75~80 years old with 'no-shows' percentage around 19% , this indicates that generally, older people 'no-shows' is less than younger people

Research Question 3 (Is there a relation between the neighbourhood and the 'no-show' rate?)

In [ ]:
```
plot5=df.neighbourhood[showed_up].value_counts()
plot6=df.neighbourhood[didnt_show_up].value_counts()

total_hood=plot5.add(plot6,fill_value=0).astype(int)

plot5_pst=plot5.divide(total_hood,fill_value=0).sort_values()
plot6_pst=plot6.divide(total_hood,fill_value=0)

fig3, (ax1, ax2) = plt.subplots(2,1,figsize=(20,16));

ax1.set_title('Relation between Neighbourhood and the number of appointments showing /
ax1.bar(plot5.index , plot5 ,label='Showed Up');
ax1.bar(plot6.index , plot6, label="Didn't Show Up");
ax1.set_ylabel('Number of Appointments',fontsize=17);
ax1.tick_params(labelrotation=90 , labelsize=10);
ax1.legend(loc=1);
#ax1.set_xticks(plot1.index)

ax2.bar(plot5_pst.index , plot5_pst, label='Showed Up');
ax2.bar(plot6_pst.index , plot6_pst, label="Didn't Show Up");
ax2.set_xlabel('Neighbourhood',fontsize=17);
ax2.set_ylabel('Percentage of poeple showing / not showing up',fontsize=17);
ax2.tick_params(labelrotation=90,labelsize=10);
#ax2.set_xticks(plot1_pst.index)
```



Q3 Observation and Conclusion

> **First plot**:there are certainly some Neighbourhoods that inquire medical attention more than others, but i can also see that as this number decreases , at some point the 'no-shows' increase.
>
> **Second plot**: we can see the percentage of the 'shows' to 'no-shows' for each Neighbourhood , Note that this plot doesnt share the same x-axis with the 1st plot.

Research Question 4 (Does the Diabatic Status, Alcoholism, Hypertension, Sending SmS & the Wellfare Program influence the 'no-show' rate?)

In [ ]:

```
total_diabetes=df.diabetes.value_counts()
show_diabetes=df[showed_up].diabetes.value_counts().divide(total_diabetes)
no_show_diabetes=df[didnt_show_up].diabetes.value_counts().divide(total_diabetes)

total_alcoholism=df.alcoholism.value_counts()
```

```
show_alcoholism=df[showed_up].alcoholism.value_counts().divide(total_alcoholism)
no_show_alcoholism=df[didnt_show_up].alcoholism.value_counts().divide(total_alcoholism)

total_hipertension=df.hipertension.value_counts()
show_hipertension=df[showed_up].hipertension.value_counts().divide(total_hipertension)
no_show_hipertension=df[didnt_show_up].hipertension.value_counts().divide(total_hiperte

total_sms_received=df.sms_received.value_counts()
show_sms_received=df[showed_up].sms_received.value_counts().divide(total_sms_received)
no_show_sms_received=df[didnt_show_up].sms_received.value_counts().divide(total_sms_rec

total_scholarship=df.scholarship.value_counts()
show_scholarship=df[showed_up].scholarship.value_counts().divide(total_scholarship)
no_show_scholarship=df[didnt_show_up].scholarship.value_counts().divide(total_scholarsh

shows=show_diabetes.append([show_alcoholism,
                            show_hipertension,
                            show_sms_received,
                            show_scholarship
                            ],ignore_index=True)

no_shows=no_show_diabetes.append([no_show_alcoholism,
                            no_show_hipertension,
                            no_show_sms_received,
                            no_show_scholarship
                            ],ignore_index=True);
```

In [ ]:

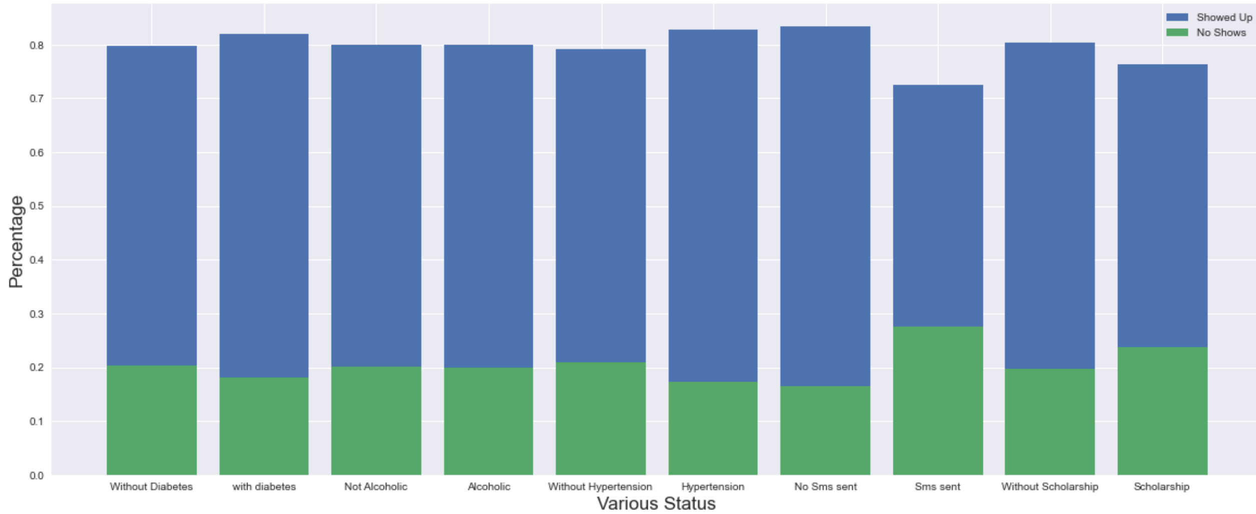
```
total_sms_received
df[showed_up].sms_received.value_counts()
```

Out[ ]:

```
0    62106
1    25698
Name: sms_received, dtype: int64
```

In [ ]:

```
plt.figure(3,figsize=(20,8))
plt.bar(shows.index,shows,label='Showed Up')
plt.bar(no_shows.index,no_shows,label='No Shows')
labels=['Without Diabetes','with diabetes','Not Alcoholic','Alcoholic','Without Hyperte
plt.xticks(shows.index,labels)
plt.xlabel('Various Status',fontsize=17)
plt.ylabel('Percentage',fontsize=17)
plt.legend();
```



Q4 Observation and Conclusion

> In the figure the blue bars indicates the % of showed up appointments while the green bars indicates the ones which didn't ,Every two successive columns are strictly indicating the status of patient i.e Not Alcholic & Alcholic.
>
> The following section will be relating to the **'no-shows'** only & not the 'shows'
>
> **Diabatic Status:** the first two bars indicates that the 'no-shows' is 20% for appointments with healthy people (out of 102,011 appointments) while its only 18% for the ones with diabetes (out of 7,893 appointments).
>
> **Alcohol Status:** the second two bars indicates that the 'no-shows' is 21% for appointments with people who don't drink alcholol (out of 106,560, appointments) while its only 19% for appointments with people who don't drink (out of 3,344 appointments).
>
> **Hypertension Status:** the third two bars indicates that the 'no-shows' is 20% for appointments with people who dont have hypertension issues (out of 88,226 appointments) while its only 17% for the ones with Hypertension (out of 21,678 appointments).
>
> **SmS Status:** the fourth two bars indicates that the 'no-shows' is 16% for appointments with people who were not notified with an SmS (out of 74,422 appointments) and **surprisingly** its 27% for appointments with the people who got notified with one (out of 35,482 appointments), intresting! , we will dig deeper into this, could it be related to Simpson's Paradox?
>
> **Welfare Program Status:** the fifth two bars indicates that the 'no-shows' is 19% for appointments with people that has no welfare program (out of 99,102 appointments) while it is 23% for the ones registered in the Bolsa Família (out of 10,802 appointments).

In my interpretation, A better way to actually see why this unexpected result happened with the Sms bars, is to see the population size and distribution for all the following groubs:

- Appointments with 'Sms sent' & 'Show up'
- Appointments with 'Sms sent' & 'No Show'
- Appointments with 'No Sms sent' & 'Show up'
- Appointments with 'No Sms sent' & 'No Show'

In [ ]:

```
recieved_sms=df.sms_received == 1
recieved_no_sms=df.sms_received == 0

sms_noshow=df[(recieved_sms & didnt_show_up)].shape[0]  #9784 appointments who got sms
                                                        #9874 /35482 = 27.8%
sms_show=df[(recieved_sms & showed_up)].shape[0]        #25698 appointments who got sms
                                                        #25698/35482 = 72.4%

nosms_noshow=df[(recieved_no_sms & didnt_show_up)].shape[0]#12,316 appointments with no
                                                           #12,316/74,422 = 16.54%
nosms_show=df[(recieved_no_sms & showed_up)].shape[0]      #62,106 appointments with  n
                                                           #62,106/74,422 =83.45%
```

In [ ]:

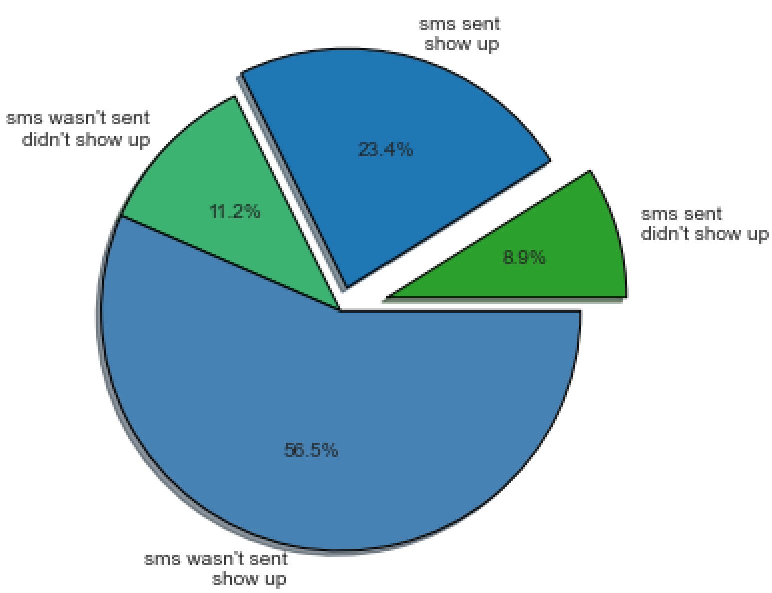
```
check =sms_noshow+sms_show+nosms_noshow+nosms_show
check
```

Out[ ]:

```
109904
```

In [ ]:

```
lbl=["sms sent\ndidn't show up","sms sent\n show up","sms wasn't sent\ndidn't show up",
myexplode = [0.2, 0.1, 0, 0]
color=['tab:green','tab:blue','mediumseagreen','steelblue']
plt.pie(x=[sms_noshow , sms_show , nosms_noshow , nosms_show] ,
        labels=lbl ,autopct='%1.1f%%' ,shadow=True ,explode=myexplode, wedgeprops={"edg
        colors=color);
```



In the bar plot above , we grouped up the percentages according to "Sms sent" "No Sms" , and demonstrated the "shows" to "no shows" to proportion of groups("Sms sent","No Sms").

But in the pie chart we can actually see the proportions of each individual group to the total sample size (109904 appointments).

Appointments who had sms sent counts 32.3% of the sample size, where Appointments with no sms sent counts 67.7% , we can also see that "show to no show" ratio is:

- for groups who were sent an sms 6:1 (*23.4/8.9*).
- for groups were No sms was sent 5:1 (*56.5/11.2*).

Eventhough the sample size of the appointments with sms sent is smaller than the one with no sms sent, we can see that by sending sms we can increase the "show to no show" ratio from 5:1 to 6:1. Turns out this was a case of Simpson's Paradox

Conclusions

General Conclusions:

Older people are more likely to show up for an appointment.
Around 31% of appointments happen in the same day registered.
Certain neighbourhoods have higher 'no show' rates than others.
Sending sms increase the "show to no show" ratio from 5:1 to 6:1

Inispiration

Using the data above, we can write a script that calculates the probabilty of someone showing up for an appointment.
this is done by receiving input form the patient from the following questions, for example:

- how old is he/she?
- which neighbourhood does he/she want to take the appointment?
- does he/she suffer from hypertension?
- does he/she suffer from diabetes?
- does he/she drink alcohol ?
- when would he/she like to come for an appointment?

Based on these answers, we would take each individual input and look up the percentage for an appointment not to show up in every single category from the pervious statistics that we calculated, for example:

- CHECK THE CODE CELL BELOW.
- Adam is **23** years old, this means he has a 25.3% chance not to show up based on his age.
- Adam wants an appointment **7 days** from now , this means he has a 26% chance not to show up based on his waiting days.

- Adam is **not suffering** from hypertension, this means he has 20.8% chance not show up based on his Hypertension status.
  .... and so on.

In [ ]:

```
print(plot4_pst[23])
print(plot2_pst[7])
print(no_show_hipertension[0])
```

```
0.2537202380952381
0.2671740024681201
0.20821526534128262
```

From this point onward, we can count only the max percentage for him, or take a weighted mean, personally i would count only the max percentage.

Limitations

One thing that i found to be strange , the fact that majority of data samples comes from scheduel day after march 2016, with earliest record at november 2015, this is a 5 months difference and thats huge , during this period there are 402 appointments only, why there wasn't more appointments during this period, i find it also weird that after march 2016 it shows significant growth for appointments. was the selection process for the sample biased?, if so this could mean that calculations above are insignificant and could be wrong.