Investigate_a_Dataset

April 20, 2021

1 Project: Investigate a Dataset (No-show appointments)

1.1 Table of Contents

Introduction

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.

Question 1: Is there relation between SMS received and miss apperment?

Question 2: Which part of the day is more appointment attendance?

Question 3: Which Age Category is more appointment attendance?

```
In [31]: # 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
         %matplotlib inline
         import requests
         import io as io
```

Data Wrangling

1.1.1 General Properties

```
In [32]: # Load the data from URL
         #df = pd.read_csv("noshowappointments-kagglev2-may-2016.csv") can not read file
         # asign url file location from the downloaded 'csv' file
         url='http://d17h27t6h515a5.cloudfront.net/topher/2017/October/59dd2e9a_noshowappointmen
```

```
        # make a requist to get url file content
        rq=requests.get(url).content
        # read the 'csv' file from the url and its decode type
        df=pd.read_csv(io.StringIO(rq.decode('utf-8')))
```

In [33]:
```
        # print out a few lines.
        df.head()
```

Out[33]:

	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 [34]:
```
        # Know no of row and columns
        df.shape
```

Out[34]: (110527, 14)

In [35]:
```
        #Perform operations to 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
```

The ouput shows that there are 110527 entries described by 14 fields. There are no null fields.

```
In [36]: #Overall Unique Columns (int64 type)
         for column in df.iloc[:,np.r_[2,5:14]].columns:
             print("{}:\n{}".format(column,df[column].unique()))
```

```
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']
```

In [37]:
```
#Look for instances of missing or possibly errant data.
#descriptive statistic
df.describe()
```

Out[37]:

	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

In [38]:
```
# count # of unique patient and appointment IDs
df.loc[:, ['PatientId', 'AppointmentID']].nunique()
```

Out[38]:
```
PatientId         62299
```

```
AppointmentID    110527
dtype: int64
```

AppointmentID is unique but PaientId is not i.e. patient may be showed more than 1

1.1.2 Dataset Field: meanning, value range, possible error data:

```
<li><b>PatienId</b>      : Patient Identification number.<BR>          (not uniqe-repeated valu
```

AppointmentID : Appointment Identification. (uniqe), may index on it rename it (AppointmentID ==> AppointmentId)

Gender : Patient gender - Male/Female (M/F).

ScheduledDay : Tells us on what day and time the patient set up their appointment. [but it is not in right format 'date'] and we need week day name to analyze on it we may get the time value and convert it to (morning /afternoon/evening) and analyze it

Age : Patient Age. (But there is -ve value '-1') If age is in age ranges it will be useful and may analyzed

AppointmentDay: Appointment date [but it is not in right format 'date']

Neighbourhood : Name of the nearest hospital location to the patient

Scholarship : Is the patient receives a scholarship? (0/1)

Hipertension : Is the patient has hypertension? (0/1)

Diabetes : Is the patient has diabetes? (0/1)

Alcoholism : Is the patient alcoholic? (0/1)

Handcap : The # of patient handicap (0,1,2,3,4) rename it Handcap ==> handicap

SMS_received : Is message sent to the patient? (0/1)

No-show : Is the patient not showed up to their appointment day? (Yes/No) ['No' mean patient showed up, 'Yes' mean patient did not show up.] we may convert its to INT to appere in all statistics and rename it (No-Show ==> No_Show)

1.1.3 Data Cleaning

```
In [39]: # After discussing the structure of the data and any problems that need to be
         #   cleaned, perform those cleaning steps in the second part of this section.
```

The youngest person '-1' and the oldest '115' are outliers in the Age field that could potentially be errors.

```
In [40]: #Remove Age <0 '-1'
         df.drop(index=df[df.Age ==-1].index, inplace=True)
```

```
In [41]: #Remove Age '115'
         df.drop(index=df[df.Age ==115].index, inplace=True)
```

```
In [42]: #change dated columes to type date
         df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
         df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])
```

```
In [43]: #Rename columns (AppointmentID, Handcap, No-show)
         df.rename(
             columns = {'AppointmentID':'AppointmentId',
                        'Handcap':'Handicap',
                        'No-show':'No_show'},
             inplace = True)

In [44]: #convert column 'No_show' to int type
         #df['No_show'] = (df.No_show == "Yes").astype(int)

In [45]: # Create a function that convert datetime (dt) to its Week Day Nname:
         day_from_datetime = lambda dt:  dt.weekday_name
         # Apply the function to the AppointmentDay column
         df['DayOfWeek'] = df.AppointmentDay.apply(day_from_datetime)

         df.DayOfWeek.value_counts().to_frame(name='Number of Appointments')

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning: `weekday_name` is
```

Out[45]:

	Number of Appointments
Wednesday	25867
Tuesday	25640
Monday	22712
Friday	19018
Thursday	17245
Saturday	39

Wednesday, and Tuesday were the most higher days for appointments, followed by Monday, Friday, and Thursday. Saturday was the lowest with 39 appointments.

```
In [46]: #create a function to get part of day from date time
         def TimeCat (Time1):
             Time1 = int(Time1.hour)
             if    Time1 <  12 :        return 'Morning'
             elif  Time1 >= 18 :        return 'Evening'
             elif  Time1 >= 12 :        return 'Afternoon'
         #create a column to show the part of day from ScheduledDay
         df['Part_day'] = df.ScheduledDay.apply(TimeCat)

In [47]: df.Part_day.value_counts().to_frame(name='Number of Appointments')
```

Out[47]:

	Number of Appointments
Morning	68476
Afternoon	40114
Evening	1931

Morning was the most higher day part for appointments, followed by Afternoon. Evening was the lowest with 1931 appointments.

```
In [48]: #create a function to age Category
         def AgeCat (Age1):
             if Age1 < 31:        return 'young'
             elif Age1 < 61 :        return 'Man'
             else:        return 'Old'
         #create a column to show age Category
         df['Age_Cat'] = df.Age.apply(AgeCat)

In [49]: df.Age_Cat.value_counts().to_frame(name='Number of Appointments')

Out[49]:        Number of Appointments
         young                   45631
         Man                     45133
         Old                     19757
```

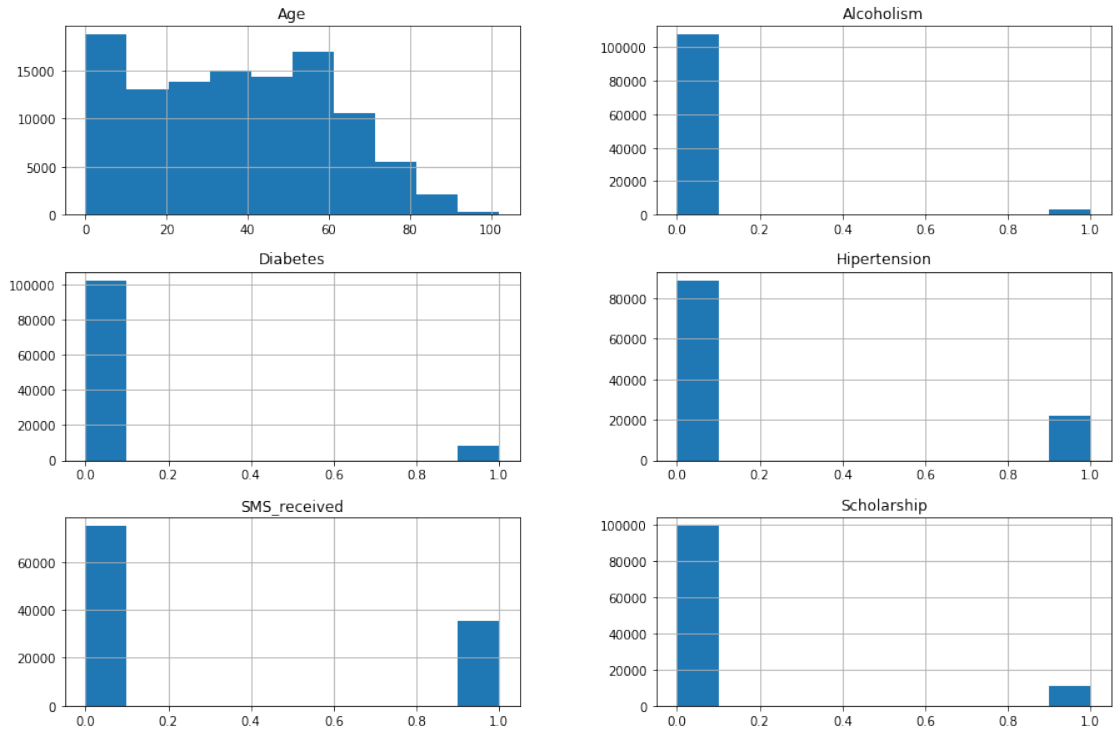
'young' was the most higher Age Category part for appointments, followed by 'Man' and 'Old' was the lowest with 19762 appointments.

```
In [50]: #df.info()
         df.head()

Out[50]:       PatientId  AppointmentId Gender        ScheduledDay AppointmentDay  Age  \
         0  2.987250e+13        5642903      F 2016-04-29 18:38:08     2016-04-29   62
         1  5.589978e+14        5642503      M 2016-04-29 16:08:27     2016-04-29   56
         2  4.262962e+12        5642549      F 2016-04-29 16:19:04     2016-04-29   62
         3  8.679512e+11        5642828      F 2016-04-29 17:29:31     2016-04-29    8
         4  8.841186e+12        5642494      F 2016-04-29 16:07:23     2016-04-29   56

                Neighbourhood  Scholarship  Hipertension  Diabetes  Alcoholism  \
         0    JARDIM DA PENHA            0             1         0           0
         1    JARDIM DA PENHA            0             0         0           0
         2      MATA DA PRAIA            0             0         0           0
         3  PONTAL DE CAMBURI            0             0         0           0
         4    JARDIM DA PENHA            0             1         1           0

            Handicap  SMS_received No_show DayOfWeek   Part_day Age_Cat
         0         0             0      No    Friday    Evening     Old
         1         0             0      No    Friday  Afternoon     Man
         2         0             0      No    Friday  Afternoon     Old
         3         0             0      No    Friday  Afternoon   young
         4         0             0      No    Friday  Afternoon     Man

In [51]: #descriptive statistic
         df.describe()

Out[51]:           PatientId  AppointmentId            Age    Scholarship  \
         count  1.105210e+05   1.105210e+05  110521.000000  110521.000000
         mean   1.474921e+14   5.675304e+06      37.085694       0.098271
         std    2.560928e+14   7.129576e+04      23.104606       0.297682
```

```
min    3.921784e+04   5.030230e+06       0.000000       0.000000
25%    4.172457e+12   5.640285e+06      18.000000       0.000000
50%    3.172598e+13   5.680569e+06      37.000000       0.000000
75%    9.438963e+13   5.725523e+06      55.000000       0.000000
max    9.999816e+14   5.790484e+06     102.000000       1.000000

        Hipertension       Diabetes     Alcoholism       Handicap  \
count  110521.000000  110521.000000  110521.000000  110521.000000
mean        0.197248       0.071869       0.030401       0.022213
std         0.397923       0.258272       0.171690       0.161440
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  110521.000000
mean        0.321034
std         0.466876
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
```

In [52]:
```
#variables frequency
df[['Age','Scholarship','Hipertension','Diabetes','Alcoholism','SMS_received','No_show'
```

Out[52]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f174939f240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1748cdb160>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f17442dc240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f174933c240>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f1748cdd940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1748cdd9e8>]], dtype=objec
```

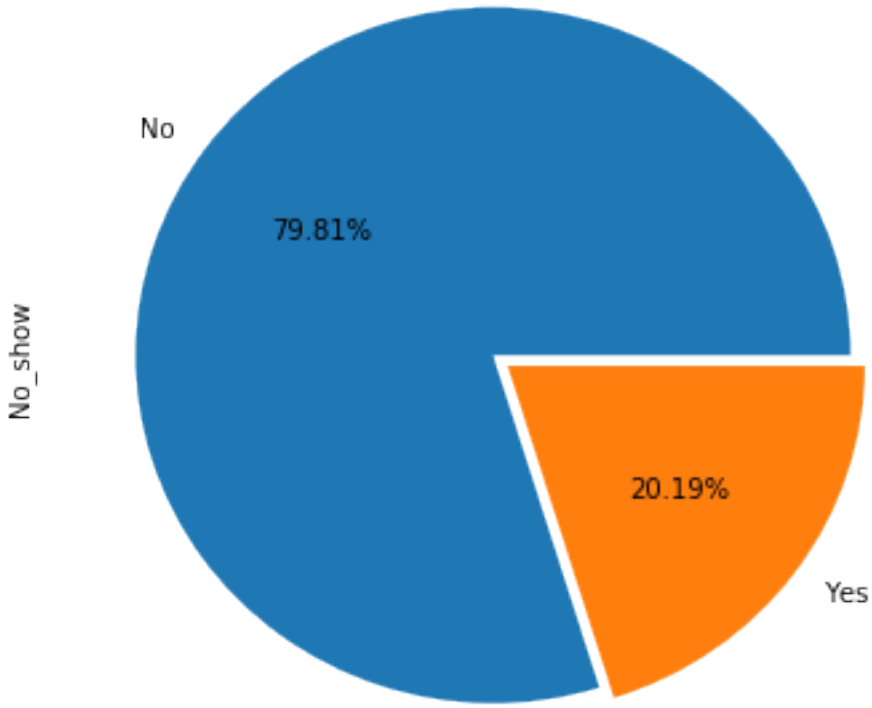


Plot comments: There is a drop down of number of Patient over 60 y, There are fewer Patient with alcoholism, diabetes, hypertension, received sms and scholarship. There are more Patient who show up for their scheduled appointment.

About 20% of all appointments resulted in no shows. The research questions will attempt to answer if the demographics of these populations are the same. They will also try to answer if this proportion is consistant given serveral priori.

Exploratory Data Analysis

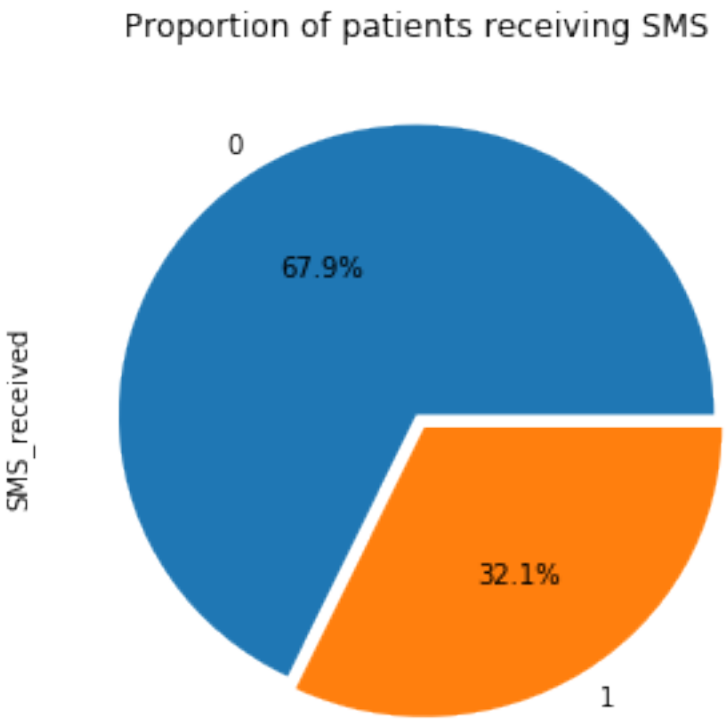
```
In [53]: #Plot the proportion of NoShows as a piechart.   Extra parameters are passed to ensure

         df.No_show.value_counts().plot.pie(figsize=(6,6), autopct='%.2f%%', explode=(0, .05))
         plt.show()
```



1.1.4 Research Question 1 (Is there relation between SMS received and miss apperment?)

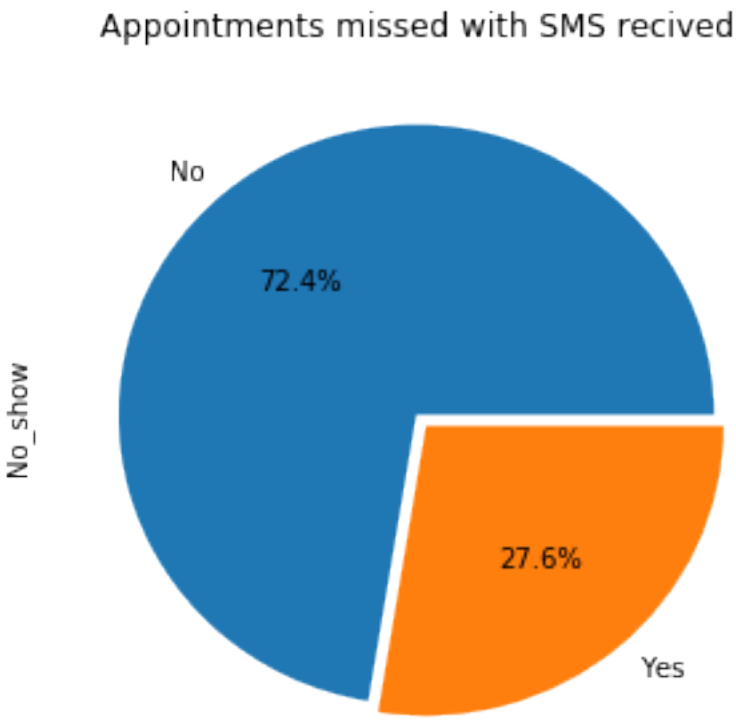
```
In [54]: df['SMS_received'].value_counts().plot(kind='pie', autopct='%1.1f%%', title='Proportion
```

```
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x7f17491fcf98>
```



About 32% of patients received SMS reminders, the question is: does receiving SMS made patients likely to attend appointments?.

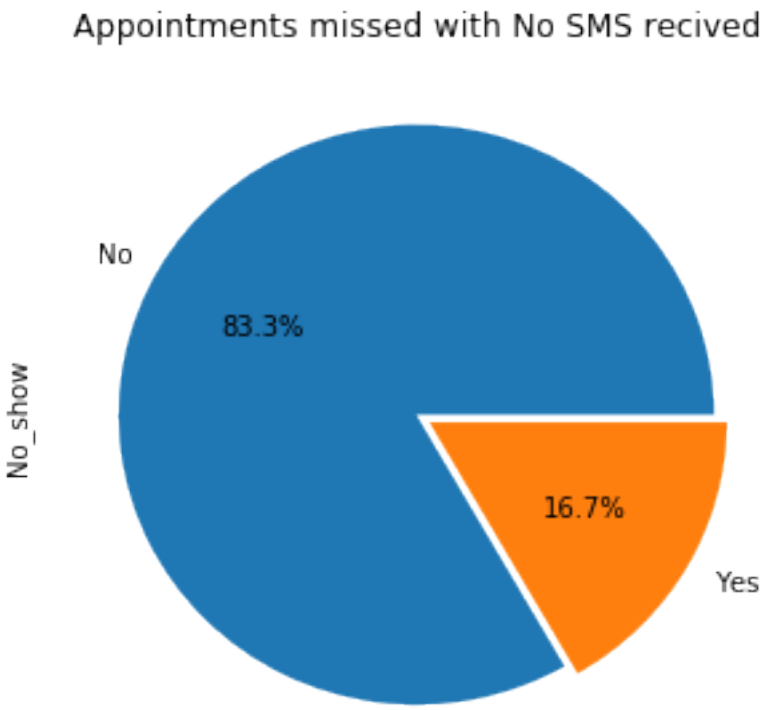
```
In [55]: # split data frame into two groups
         SMS = df.query('SMS_received == 1')
         NoSMS = df.query('SMS_received == 0')

In [56]: # what proportion of appointments were missed by those who received SMS?
         SMS['No_show'].value_counts().plot(kind='pie', title='Appointments missed with SMS reci

Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1747a30940>
```



About 27% of patients received SMS reminders and missed appointments

```
In [57]: # what proportion of appointments were missed by those who did not received SMS?
         NoSMS['No_show'].value_counts().plot(kind='pie', title='Appointments missed with No SMS
```

```
Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1747437438>
```



About 17% of patients does not receive SMS reminders and missed appointments

```
In [58]: # Use this, and more code cells, to explore your data. Don't forget to add
         #   Markdown cells to document your observations and findings.
```

1.1.5 Research Question 2 (Which part of the day is more appointment attendance?)

```
In [59]: df.Part_day.value_counts().to_frame(name='Number of Appointments')
```

```
Out[59]:            Number of Appointments
         Morning                     68476
         Afternoon                   40114
         Evening                      1931
```

Morning was the most higher day part for appointments, followed by Afternoon. Evening was the lowest with 1931 appointments.

```
In [60]: #
         df.Part_day[Show].value_counts().plot(kind='bar', alpha=0.5,color='red', label='Show')
         df.Part_day[noshows].value_counts().plot(kind='bar', alpha=0.5, color='blue',label='No-
         plt.legend()
         plt.title('Day Parts Appointment Attendance')
         plt.xlabel('Day Parts')
```

```
plt.ylabel('Patient Quantity')
plt.xticks(rotation=0)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-60-96805a3435a9> in <module>()
  1 #
----> 2 df.Part_day[Show].value_counts().plot(kind='bar', alpha=0.5,color='red', label='Show
  3 df.Part_day[noshows].value_counts().plot(kind='bar', alpha=0.5, color='blue',label='
  4 plt.legend()
  5 plt.title('Day Parts Appointment Attendance')

NameError: name 'Show' is not defined
```

the above graph show the comparing between Attendance (Show) and Non attendence (No Show) the next 3 graph show the % of No Show in each Day Part

```
In [ ]: # split data frame into 3 groups
        Morning = df.query('Part_day == "Morning"')
        Afternoon = df.query('Part_day == "Afternoon"')
        Evening = df.query('Part_day == "Evening"')
```

```
In [ ]: Morning['No_show'].value_counts().plot(kind='pie', title='Appointments missed in Morning
```

the attendence (Show) in the morning is about 81% to no attendence (NO SHow) 19%

```
In [ ]: Afternoon['No_show'].value_counts().plot(kind='pie', title='Appointments missed in After
```

the attendence (Show) in the Afternoon is about 78% to no attendence (NO SHow) about 22%

```
In [ ]: Evening['No_show'].value_counts().plot(kind='pie', title='Appointments missed in Evening
```

the attendence (Show) in the Evening is about 78% to no attendence (NO SHow) about 22%

1.1.6 Research Question 3 (Which Age Category is more appointment attendance?)

```
In [ ]: Show = (df.No_show == 'No')
        noshows = (df.No_show == 'Yes')
        df[Show].Age.plot.hist()
        df[noshows ].Age.plot.hist()
        plt.legend(['Show Apointments', 'No Shows'])
        plt.show()
        print('Show Appointments Mean Age:{:.2f}'.format(df[Show].Age.mean()))
        print('Missed Appointments Mean Age:{:.2f}'.format(df[noshows].Age.mean()))
```

This histogram overlays the distributions of Show appointments and Mised (No Show) so that they can be compared. The peak near 60 years is much smaller for no shows. The mean is also lower but is not apparent with a histogram.

So the vast majority of patients are 60 or under. Let's divide our patient population into three groups: young (under 30), Man (between 30 and less than 60) and old (60 and over).

```
In [ ]: df.Age_Cat.value_counts().to_frame(name='Number of Appointments')
```

'young' was the most higher Age Category part for appointments, followed by 'Man' and 'Old' was the lowest with 19762 appointments.

Then we will see if there are differences in the rate at which these groups miss appointments.

```
In [ ]: young = df.query('Age < 30')
        Man   = df.query('Age >= 30 & Age < 60')
        old   = df.query('Age >= 60')
```

```
In [ ]: young['No_show'].value_counts().plot(kind='pie', autopct='%1.1f%%', figsize=[5,5], title
```

the attendence (Show) in the Young is about 77% to no attendence (NO SHow) about 23%

```
In [ ]: Man['No_show'].value_counts().plot(kind='pie', autopct='%1.1f%%', figsize=[5,5], title='
```

the attendence (Show) in the Man is about 80% to no attendence (NO SHow) about 20%

```
In [ ]: old['No_show'].value_counts().plot(kind='pie', autopct='%1.1f%%', figsize=[5,5], title='
```

the attendence (Show) in the Old is about 85% to no attendence (NO SHow) about 15%

Conclusions

Our exploration has generated the statistics and graphs allowing us to fulfill the purposes of our investigation. We can see how certain factors affect the likelihood of patients missing their appointments.

Is there relation between SMS received and miss apperment? Only around 32% of patients received an SMS reminder, and to our surprise those who did receive it were more likely to miss an appointment (27.6%) than those who didn't (16.7%). Perhaps we should consider no longer sending SMS reminders.

Which part of the day is more appointment attendance? Morning was the most higher day part for appointments, followed by Afternoon. Evening was the lowest with 1931 appointments, the attendence (Show) in the morning is about 81% to no attendence (NO SHow) 19%, the attendence (Show) in the Afternoon is about 78% to no attendence (NO SHow) about 22%, the attendence (Show) in the Evening is about 78% to no attendence (NO SHow) about 22%. The data suggests that the morning is higher day part, so morning need enough doctor's appointments.

Which Age Category is more appointment attendance? Though the vast majority of patients are young (under 30) or Man (between 30 and 60), young people were most likely to miss appointments (22.9%), Old people over 60 were least likely to miss appointments (15.3%), and Man people were in between (19.8%). The data suggests that the older people get, the more diligent they are about keeping doctor's appointments.

Limitations

The are many limitations in the Data:

```
<li>Most data set variables are categorical , which does not allow a high level of statistical m
```

The statistics used here are descriptive statistics, not inferential, meaning that any hypotheses or controlled experiments or Inferences can not created With data.

Lot of details for certain factors are not found to draw conclusions. For the SMS received example, the date and time of sending or receiveing SMS that give us that SMS is importent criteria or just neglect it.

In the Age field, we have error like out of range and outliers values which could potentially give low accurate result.

```
In [61]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[61]: 0
```