investigate-a-dataset-template

March 19, 2020

1 Project: No Show Appointments

1.0.1 Table of contents

- Introduction
- Data Wrangling
- Exploratory Data Analysis
- Conclusions
- Links

Introduction

The structure of the dataset There are a little more than 110k appointments information which are divided into 14 columns. These columns provide data about the patient, their health issues and whether or not they attended their appointment.

What are the main features of interest in the dataset? I am mostly interested in finding out the reasons of why a patient might miss their appointment and which factors inluence those events. Also, what were the different age groups and health issues as well as their distributions

Data Wrangling

1.0.2 Gather

[1]:
```
# Import necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import datetime

%matplotlib inline
```

[2]:
```
# Load the dataset
appointments = pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
```

1.0.3 Assess and note issues

[3]:
```
appointments.head()
```

[3]:
```
      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

         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

   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
```

[4]:
```
appointments.shape
```

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

[5]:
```
appointments.dtypes
```

[5]:
```
PatientId         float64
AppointmentID       int64
Gender             object
ScheduledDay       object
AppointmentDay     object
Age                 int64
Neighbourhood      object
Scholarship         int64
Hipertension        int64
Diabetes            int64
Alcoholism          int64
Handcap             int64
SMS_received        int64
No-show            object
dtype: object
```

[6]:
```
appointments.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
```

[7]:
```
appointments.nunique()
```

[7]:
```
PatientId          62299
AppointmentID     110527
Gender                 2
ScheduledDay      103549
AppointmentDay        27
Age                  104
Neighbourhood         81
Scholarship            2
Hipertension           2
Diabetes               2
Alcoholism             2
Handcap                5
SMS_received           2
No-show                2
dtype: int64
```

[8]:
```
appointments.Handcap.value_counts()
```

[8]:
```
0    108286
1      2042
2       183
3        13
4         3
```

```
Name: Handcap, dtype: int64
```

[9]:
```
appointments.Age.value_counts()
```

[9]:
```
0      3539
1      2273
52     1746
49     1652
53     1651
       …
115       5
100       4
102       2
99        1
-1        1
Name: Age, Length: 104, dtype: int64
```

[10]:
```
appointments.Neighbourhood.value_counts()
```

[10]:
```
JARDIM CAMBURI                 7717
MARIA ORTIZ                    5805
RESISTÊNCIA                    4431
JARDIM DA PENHA                3877
ITARARÉ                        3514
                               …
ILHA DO BOI                      35
ILHA DO FRADE                    10
AEROPORTO                         8
ILHAS OCEÂNICAS DE TRINDADE       2
PARQUE INDUSTRIAL                 1
Name: Neighbourhood, Length: 81, dtype: int64
```

[11]:
```
appointments.describe()
```

[11]:
```
          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
```

1.0.4 Issues

- uppercase column names
- incorrectly spelled column names (Hipertension, Handcap)
- patientid, appointmentid, AppoinmentDay, ScheduledDay, no-show are not consistent with SMS_recieved
- incorrect patient id dtype
- scheduleday and appointmentday columns are strings
- incorrect gender dtype
- confusing values for No-show column
- other numbers (2, 3, 4) are given for True (1) or False (0) for Handicap column
- negative number (-1) for age
- raw integer values for age are not very useful to see different trends
- the placement of issue types (hypertension, diabetes, alcoholism, handicap) makes visualisations difficult
- surprising appointment days. There seems to be only 27 unique dates for appointments. This issue will be addressed after converting schedule and appointment columns to datetime

1.0.5 Data Cleaning

[12]:
```
# Make a copy of the dataset
copy = appointments.copy()
```

Uppercase column namese

[13]:
```
# Turn all column names to lowercase
appointments.columns = appointments.columns.str.lower()
# Test
appointments.head()
```

[13]:
```
       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

         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

   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
```

Incorrectly spelled column names (Hipertension, Handcap)

[14]:
```
# Rename the above columns
appointments.rename(columns={'hipertension': 'hypertension',
                             'handcap': 'handicap'}, inplace=True)
# Test
appointments.columns
```

[14]:
```
Index(['patientid', 'appointmentid', 'gender', 'scheduledday',
       'appointmentday', 'age', 'neighbourhood', 'scholarship', 'hypertension',
       'diabetes', 'alcoholism', 'handicap', 'sms_received', 'no-show'],
      dtype='object')
```

Patientid, appointmentid, AppoinmentDay, ScheduledDay, no-show are not consistent with SMS_recieved

[15]:
```
# Rename the above columns
appointments.rename(columns={'appointmentday': 'appointment_day',
 'scheduledday': 'scheduled_day', 'patientid': 'patient_id',
    'appointmentid': 'appointment_id', 'no-show': 'no_show'}, inplace=True)
# Test
appointments.columns
```

[15]:
```
Index(['patient_id', 'appointment_id', 'gender', 'scheduled_day',
       'appointment_day', 'age', 'neighbourhood', 'scholarship',
       'hypertension', 'diabetes', 'alcoholism', 'handicap', 'sms_received',
       'no_show'],
```

```
       dtype='object')
```

Other numbers are given for True (1) or False (0) for Handicap column - 2, 3, 4

[16]:
```
# Replace all the values with ones if not 0
appointments.handicap = appointments.handicap.apply(
            lambda x: 1 if x != 0 else x).astype(dtype=np.int64)

# Test
appointments.handicap.value_counts()
```

[16]:
```
0    108286
1      2241
Name: handicap, dtype: int64
```

Raw integer values are not very useful to see different trends I will create a custom function to divide the age column into different age groups

[17]:
```
def group_age(age):
    """Groups the ages into 6 different groups:
    Children, Teenagers, Yound Adults, Middle Age Adults, Elderly, over 90"""
    if 0 <= age < 12:
        return 'children'
    elif 12 <= age < 19:
        return 'teenagers'
    elif 19 <= age < 35:
        return 'young adults'
    elif 35 <= age < 60:
        return 'middle age adults'
    elif 60 <= age < 90:
        return 'elderly'
    else:
        return 'over 90'
```

[18]:
```
# Run the above function
appointments['age_groups'] = appointments.age.apply(group_age)

# Test
appointments.sample(10)
```

[18]:

	patient_id	appointment_id	gender	scheduled_day	\
40272	3.511322e+12	5645689	F	2016-05-02T10:15:35Z	
28292	5.727454e+11	5580130	M	2016-04-13T16:41:27Z	
35701	8.896946e+13	5653559	M	2016-05-03T12:37:51Z	
21689	1.179523e+13	5642963	F	2016-05-02T06:53:30Z	
78663	5.628478e+13	5679523	F	2016-05-10T10:03:31Z	
22053	3.539633e+14	5685556	M	2016-05-11T10:36:23Z	

```
33773  5.721124e+12         5613454      F  2016-04-25T09:19:52Z
61818  9.458326e+13         5701177      M  2016-05-16T10:12:31Z
9392   6.834123e+13         5668528      F  2016-05-06T09:15:16Z
93883  1.213915e+11         5754485      M  2016-05-31T16:21:29Z

           appointment_day  age        neighbourhood  scholarship  \
40272  2016-05-02T00:00:00Z   55         SANTA MARTHA            0
28292  2016-05-12T00:00:00Z   81        MATA DA PRAIA            0
35701  2016-05-04T00:00:00Z   34  ILHA DE SANTA MARIA            0
21689  2016-05-04T00:00:00Z   54       NOVA PALESTINA            0
78663  2016-05-10T00:00:00Z   49           ANDORINHAS            0
22053  2016-05-11T00:00:00Z   53            DO QUADRO            0
33773  2016-05-04T00:00:00Z   59       BENTO FERREIRA            0
61818  2016-05-19T00:00:00Z   64         PRAIA DO SUÁ            0
9392   2016-05-10T00:00:00Z   23        SANTOS DUMONT            0
93883  2016-06-06T00:00:00Z    2      JARDIM DA PENHA            0

       hypertension  diabetes  alcoholism  handicap  sms_received no_show  \
40272             0         0           0         0             0      No
28292             1         0           0         0             1      No
35701             0         0           0         0             0     Yes
21689             1         1           0         0             0     Yes
78663             1         0           0         0             0      No
22053             0         0           0         0             0      No
33773             1         0           0         0             1      No
61818             0         0           0         0             0      No
9392              0         0           0         0             1     Yes
93883             0         0           0         0             1      No

             age_groups
40272  middle age adults
28292           elderly
35701      young adults
21689  middle age adults
78663  middle age adults
22053  middle age adults
33773  middle age adults
61818           elderly
9392       young adults
93883           children
```

The placement of issue types (hypertension, diabetes, alcoholism, handicap) makes visualisations difficult

[19]:
```
def group_issues(df):
    """Assigns an issue label for every patient who had an appointment.
       If no or multiple issues, the function returns "none" or
```

```
        "multiple" accordingly
    """
    # Create a new column for the new variable
    df['issue'] = (df.hypertension + df.diabetes + df.alcoholism + df.handicap).
↪astype(np.int64)

    # Convert 0s to 'none' and values greater than 1 to 'multiple'
    df.issue = df.issue.apply(lambda x: 'none' if x == 0 else 'multiple' if x >␣
↪1 else x)

    # Convert the remaining records (patiens with only one issue)
    # to issue name
    for i in range(df.shape[0]):
        if df.iloc[i].issue == 1 and df.iloc[i].hypertension == 1:
            df.loc[i, 'issue'] = 'hypertension'
        elif df.iloc[i].issue == 1 and df.iloc[i].diabetes == 1:
            df.loc[i, 'issue'] = 'diabetes'
        elif df.iloc[i].issue == 1 and df.iloc[i].alcoholism == 1:
            df.loc[i, 'issue'] = 'alcoholism'
        elif df.iloc[i].issue == 1 and df.iloc[i].handicap == 1:
            df.loc[i, 'issue'] = 'handicap'
```

[20]:
```
# Run the above function on the dataframe to get issue groups
group_issues(appointments)
```

[21]:
```
# Test
appointments.groupby('issue').issue.value_counts()
```

[21]:
```
issue          issue
alcoholism     alcoholism       1922
diabetes       diabetes         1341
handicap       handicap         1197
hypertension   hypertension    13663
multiple       multiple         8289
none           none            84115
Name: issue, dtype: int64
```

Incorrect patient id dtype

[22]:
```
# Set the datatype of patient id to int
# I used string manipulation to convert patient ids from scientific notation to
# string, then got rid of all decimals. If works correctly, it still should give
# us the same number of unique patient ids
appointments.patient_id = appointments.patient_id.astype(str).str.strip('.0').
↪str.replace('.', '').astype(dtype=np.int64)

# Test
```

```
appointments.patient_id.dtype, appointments.patient_id.nunique()
```

[22]:
```
(dtype('int64'), 62299)
```

Scheduled_day and appointment_day columns are strings

[23]:
```
# Convert columns to datetime
appointments['scheduled_day'] = pd.to_datetime(appointments['scheduled_day'])
appointments['appointment_day'] = pd.
 ↪to_datetime(appointments['appointment_day'])

# Test
appointments.dtypes
```

[23]:
```
patient_id                      int64
appointment_id                  int64
gender                         object
scheduled_day      datetime64[ns, UTC]
appointment_day    datetime64[ns, UTC]
age                             int64
neighbourhood                  object
scholarship                     int64
hypertension                    int64
diabetes                        int64
alcoholism                      int64
handicap                        int64
sms_received                    int64
no_show                        object
age_groups                     object
issue                          object
dtype: object
```

Incorrect gender dtype

[24]:
```
# Convert gender column to categorical dtype
appointments.gender = appointments.gender.astype(dtype='category')

# Test
appointments.gender.dtype
```

[24]:
```
CategoricalDtype(categories=['F', 'M'], ordered=False)
```

String values for no_show column

[25]:
```
# First, this column uses yes or no in a confusing manner. Let's fix that
# Also, change the column name to a more intuitive one
appointments.no_show = appointments.no_show.apply(
                                lambda x: "Yes" if x == 'No' else 'No')
```

```
appointments.rename(columns={'no_show': 'attended'}, inplace=True)

# Now, convert them to 0s and 1s
appointments.attended = appointments.attended.apply(
                        lambda x: 1 if x == 'Yes' else 0)

# Test
appointments.attended.value_counts()
```

[25]:
```
1    88208
0    22319
Name: attended, dtype: int64
```

Negative number (-1) for age

[26]:
```
# Drop the row with negative age
negative_index = appointments[appointments.age == -1].index
appointments.drop(negative_index, inplace=True)

# Test
appointments.age.value_counts()
```

[26]:
```
0      3539
1      2273
52     1746
49     1652
53     1651
       …
98        6
115       5
100       4
102       2
99        1
Name: age, Length: 103, dtype: int64
```

Surprising appointment days. There seems to be only 27 unique dates for appointments.

[27]:
```
(appointments.appointment_day - appointments.scheduled_day).astype(str).str.
↪contains('-').sum()
```

[27]:
```
38567
```

There seems to be almost 39k appointment records which had schedule day later than appointment day which does not make sense (patients should first schedule, then come to appointments). Therefore, I am going look at all of the records with appointment day happening before schedule day.

[28]:
```
appointments[appointments.appointment_day < appointments.scheduled_day]
```

[28]:

```
             patient_id  appointment_id gender              scheduled_day  \
0         29872499824296         5642903      F 2016-04-29 18:38:08+00:00
1        558997776694438         5642503      M 2016-04-29 16:08:27+00:00
2          4262962299951         5642549      F 2016-04-29 16:19:04+00:00
3           867951213174         5642828      F 2016-04-29 17:29:31+00:00
4          8841186448183         5642494      F 2016-04-29 16:07:23+00:00
…                    …               …    …                         …
110511      823599626588         5786742      F 2016-06-08 08:50:20+00:00
110512    98762456447375         5786368      F 2016-06-08 08:20:01+00:00
110513    86747784995281         5785964      M 2016-06-08 07:52:55+00:00
110514     2695685177138         5786567      F 2016-06-08 08:35:31+00:00
110517     5574942418928         5780122      F 2016-06-07 07:38:34+00:00

                  appointment_day  age      neighbourhood  scholarship  \
0       2016-04-29 00:00:00+00:00   62    JARDIM DA PENHA            0
1       2016-04-29 00:00:00+00:00   56    JARDIM DA PENHA            0
2       2016-04-29 00:00:00+00:00   62      MATA DA PRAIA            0
3       2016-04-29 00:00:00+00:00    8  PONTAL DE CAMBURI            0
4       2016-04-29 00:00:00+00:00   56    JARDIM DA PENHA            0
…                             …  …                 …          …
110511  2016-06-08 00:00:00+00:00   14        MARIA ORTIZ            0
110512  2016-06-08 00:00:00+00:00   41        MARIA ORTIZ            0
110513  2016-06-08 00:00:00+00:00    2    ANTÔNIO HONÓRIO            0
110514  2016-06-08 00:00:00+00:00   58        MARIA ORTIZ            0
110517  2016-06-07 00:00:00+00:00   19        MARIA ORTIZ            0

        hypertension  diabetes  alcoholism  handicap  sms_received  attended  \
0                  1         0           0         0             0         1
1                  0         0           0         0             0         1
2                  0         0           0         0             0         1
3                  0         0           0         0             0         1
4                  1         1           0         0             0         1
…                …         …           …         …             …         …
110511             0         0           0         0             0         1
110512             0         0           0         0             0         1
110513             0         0           0         0             0         1
110514             0         0           0         0             0         1
110517             0         0           0         0             0         1

               age_groups         issue
0                  elderly  hypertension
1       middle age adults          none
2                  elderly          none
3                 children          none
4       middle age adults      multiple
```

```
…                    …           …
110511          teenagers          none
110512  middle age adults          none
110513           children          none
110514  middle age adults          none
110517       young adults          none

[38567 rows x 16 columns]
```

As can be seen from above, there were patients who attended their appointments but had their appointments earlier than their schedule day. So, we can only assume that it was just some mistake when collecting the data, but we have to make sure. Before dropping these records, just to be safe, I am going to leave out the records which had -1 day difference, because that could mean the appointment might have happened on the schedule day (even though there is a negative hour difference)

[29]:
```
appointments['difference'] = appointments.appointment_day - appointments.
↪scheduled_day
```

[30]:
```
problematic_appoints = appointments[appointments.difference < datetime.
↪timedelta(days=-1)]
problematic_appoints
```

[30]:
```
            patient_id  appointment_id gender              scheduled_day  \
27033    7839272661752         5679978      M 2016-05-10 10:51:53+00:00
55226    7896293967868         5715660      F 2016-05-18 14:50:41+00:00
64175   24252258389979         5664962      F 2016-05-05 13:43:58+00:00
71533  998231581612122         5686628      F 2016-05-11 13:49:20+00:00
72362    3787481966821         5655637      M 2016-05-04 06:50:57+00:00

                appointment_day  age  neighbourhood  scholarship  \
27033 2016-05-09 00:00:00+00:00   38    RESISTÊNCIA            0
55226 2016-05-17 00:00:00+00:00   19  SANTO ANTÔNIO            0
64175 2016-05-04 00:00:00+00:00   22     CONSOLAÇÃO            0
71533 2016-05-05 00:00:00+00:00   81  SANTO ANTÔNIO            0
72362 2016-05-03 00:00:00+00:00    7     TABUAZEIRO            0

       hypertension  diabetes  alcoholism  handicap  sms_received  attended  \
27033             0         0           0         1             0         0
55226             0         0           0         1             0         0
64175             0         0           0         0             0         0
71533             0         0           0         0             0         0
72362             0         0           0         0             0         0

              age_groups     issue        difference
27033  middle age adults  handicap -2 days +13:08:07
55226       young adults  handicap -2 days +09:09:19
```

```
64175       young adults      none -2 days +10:16:02
71533            elderly      none -7 days +10:10:40
72362           children      none -2 days +17:09:03
```

Now we have 5 patients which has the same issue as above but also, did not attend their appointment. I drop those:

[31]:
```
appointments.drop(problematic_appoints.index, inplace=True)

# Test
appointments.shape
```

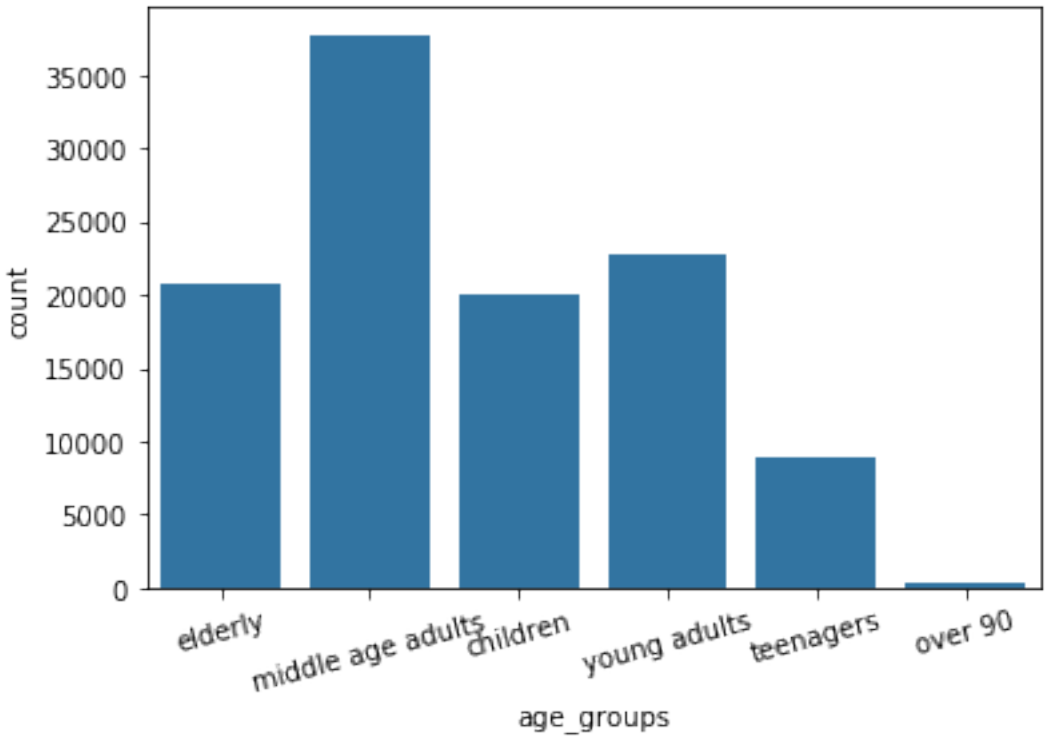
[31]:
```
(110521, 17)
```

[32]:
```
# Drop the difference column since we do not need it anymore
appointments.drop('difference', axis=1, inplace=True)
```

Exploratory Data Analysis

First, I am going to look at the distribution of age groups

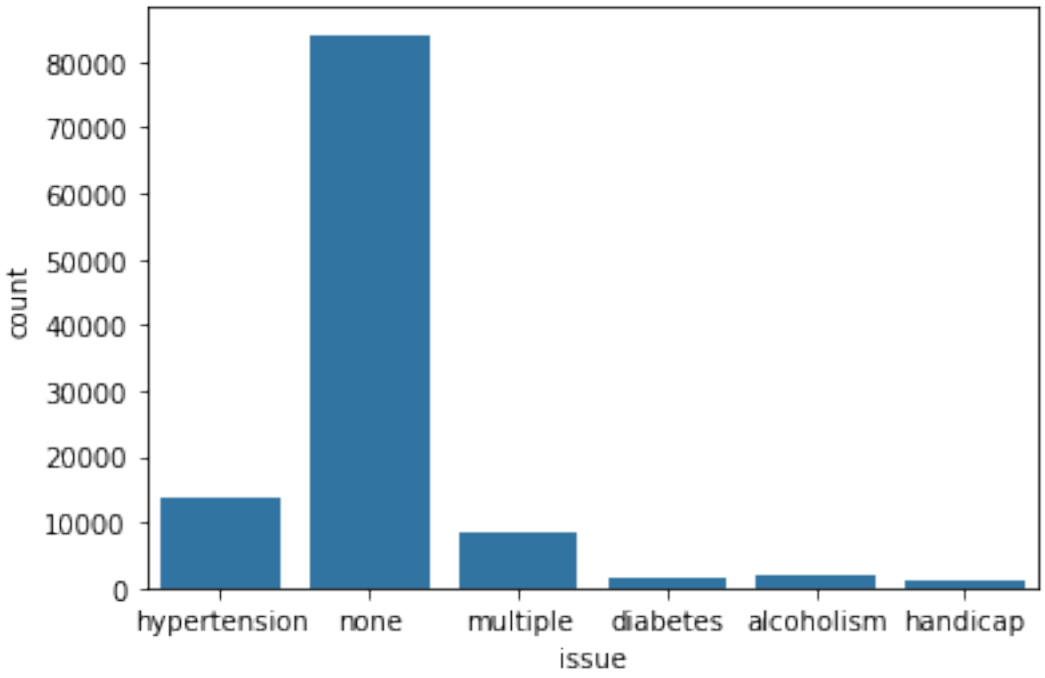
[33]:
```
base_color = sb.color_palette()[0]
sb.countplot(data=appointments, x='age_groups', color=base_color)
plt.xticks(rotation=15);
```



It is clear from above that middle aged adults represent the majority of patients, while there are very few people over 90s

Now, let's look at the distribution of different issues.

[34]:
```
sb.countplot(data=appointments, x='issue', color=base_color);
```



The above countplot reveals that there were much more patients who did not have any issues when they scheduled. Among patients, hypertension seems to be the most common disease, followed by patients who have multiple health issues.

I expect that the two of the main reasons of why patients missed their appointments are age and their health issues. After I drop columns which I no longer have use for, I will start comparing patients who missed and did not miss appointments agains those two variables.

[35]:
```
appointments.drop(['appointment_id', 'scheduled_day',
                   'appointment_day', 'age', 'neighbourhood', 'hypertension',
                  'diabetes', 'alcoholism', 'handicap', 'gender'], axis=1,␣
 ↪inplace=True)
```

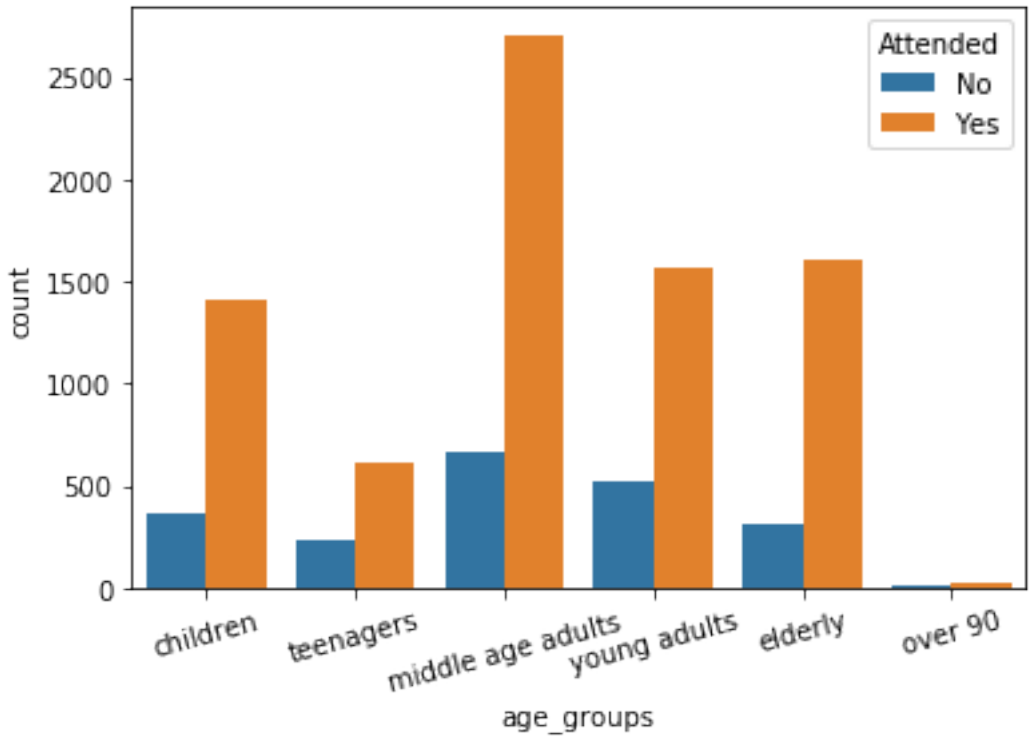
[36]:
```
# First look at the overall proportion of people who attended their appointments
overall_attendance_prop = appointments.attended.mean()
overall_attendance_prop
```

[36]:
```
0.7981017182254956
```

[37]:
```
# Take a sample from the datababse to reduce computation time
subset_appointments = appointments.sample(10000, replace=False)
```

[38]:
```
# Now create masks for attendance
attended = subset_appointments.attended == True
not_attended = subset_appointments.attended == False
```

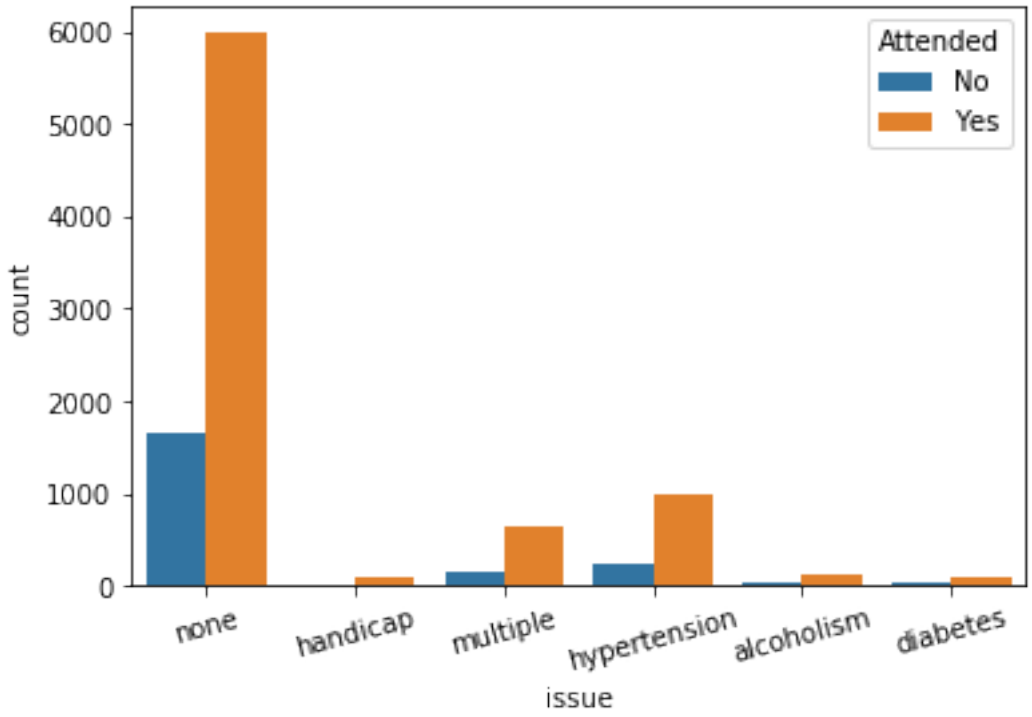
[39]:
```
# Plot the difference in attendance between age groups
sb.countplot(data=subset_appointments, x='age_groups', hue='attended')
plt.xticks(rotation=15)
plt.legend(labels=['No', 'Yes'], title='Attended');
```



From earlier plots we saw that the majority of patients were middle aged adults. Therefore, the last diagram makes sense as to why there are more middle aged adults for both attendance categories.

Next I am going to look at the disribution of proportions of health issues

[40]:
```
sb.countplot(data=subset_appointments, x='issue', hue='attended')
plt.xticks(rotation=15)
plt.legend(labels=['No', 'Yes'], title='Attended');
```



The above plots did not reveal any signs that health issues or age affect missing appointments. Even though, middle aged adults and patients who had no health issues had higher figures for attendance, that is largely due to the fact that those groups represent the majority of the records. So, we will dive deeper. In order to answer my main question, I have to find the proportions of attendance for every age group and issue.

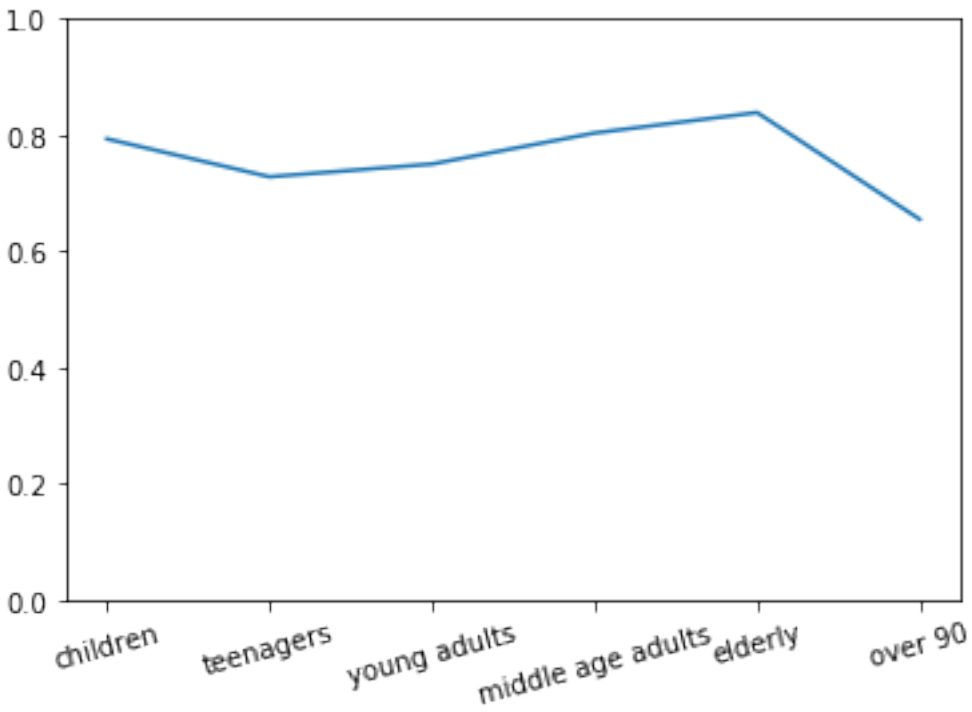
[41]:
```
def find_props(df):
    """
    Purpose:
    Finds the proportion of attendance for every age, issue group
    and maps the results to a dictionary

    Returns: A tuple of Dictionaries
    """
    age_groups = ['children', 'teenagers', 'young adults', 'middle age adults',␣
↪'elderly', 'over 90']
    issues = ['hypertension', 'diabetes', 'alcoholism', 'handicap', 'none',␣
↪'multiple']
    age_group_dict = {}
    issue_group_dict = {}
    for column in ['age_groups', 'issue']:
        if column == 'age_groups':
            for group in age_groups:
```

```
                age_group_dict[group] = df[attended][df[attended][column] ==␣
↪group][column].shape[0] / df[df[column] == group][column].shape[0]
        if column == 'issue':
            for group in issues:
                issue_group_dict[group] = df[attended][df[attended][column] ==␣
↪group][column].shape[0] / df[df[column] == group][column].shape[0]
    return age_group_dict, issue_group_dict
```

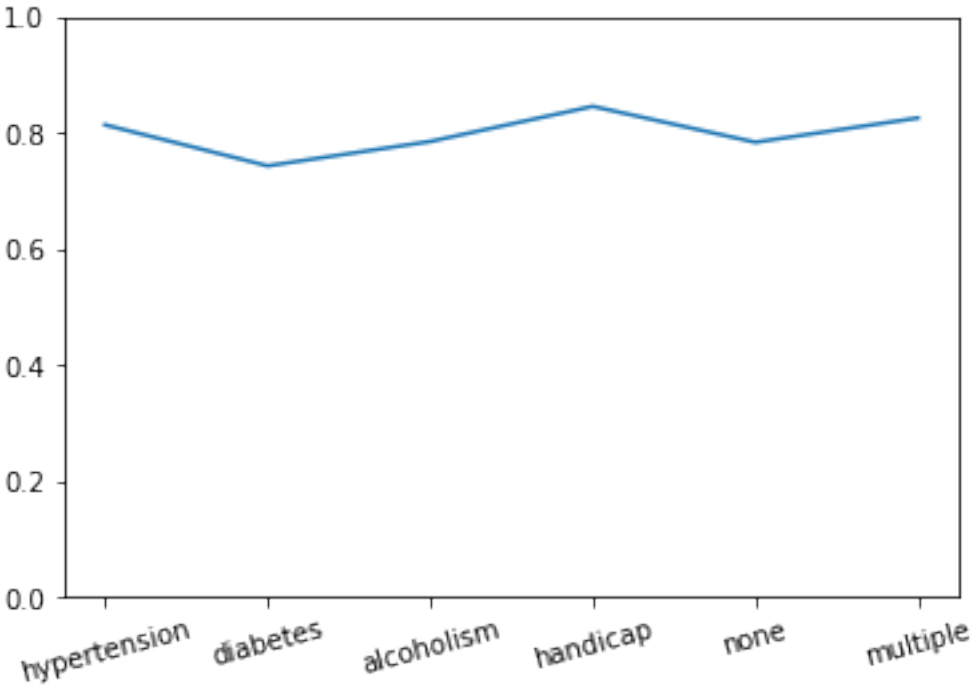
[42]:
```
age, issue = find_props(subset_appointments)
age, issue
```

[42]:
```
({'children': 0.7934537246049661,
  'teenagers': 0.7281437125748503,
  'young adults': 0.7501199040767386,
  'middle age adults': 0.8035025230038587,
  'elderly': 0.8387434554973822,
  'over 90': 0.6551724137931034},
 {'hypertension': 0.813488759367194,
  'diabetes': 0.7428571428571429,
  'alcoholism': 0.7848101265822784,
  'handicap': 0.845360824742268,
  'none': 0.7835011112563733,
  'multiple': 0.8253164556962025})
```

[43]:
```
plt.plot(pd.Series(age))
plt.ylim((0, 1))
plt.xticks(rotation=15);
```



[44]:
```
plt.plot(pd.Series(issue))
plt.ylim((0, 1))
plt.xticks(rotation=15);
```



[46]:
```
# Save the dataset to a csv file
appointments.to_csv('appointments_master.csv')
```

Conclusions

After all the plotting and data wrangling, it appears that there is not one single factor which influences whether a patient comes to the appointment or not. In age groups, over 90 group seems to be the most common to miss their appointment. Similarly, handicapped people were more likely to miss their appointments. However, the differences between proportions were very negligible and therefore, they are not enough to draw any insight as to why a patient did not attend their appointment.

Links - GitHub - Dataset original source - Kaggle