IST-652
Mini Project-1

Exploratory Data Analysis of Medical Appointments Dataset using Python

By:
Nivesh Vaze
nvvaze@syr.edu
411946624

Contents

Introduction

Let's understand the concept of Exploratory Data Analysis.

Exploratory Data Analysis is the process of investigating the dataset to discover patterns, and anomalies (outliers), and form hypotheses based on our understanding of the dataset. EDA involves extracting data, cleaning, and transforming data for generating summary statistics of numerical data in the dataset and creating various graphical representations to understand the data in a better way. To summarize, EDA helps to generate hypotheses about data, detect its anomalies and reveal the structure.

The IST 652 Mini Project-1 is my first attempt to explore a Medical Appointment No-Show dataset using Python. The Objective is to understand the contributing factors for missing appointments. This report provides the summary as well as result of all the analysis performed in this project.

Dataset and Source

We will be working with the Medical Appointment No-Show dataset that contains information about the patients' appointments.

Each patient's record is characterized by the following features:

- **PatientID** — a unique identifier of a patient
- **AppointmentID** — a unique identifier of an appointment
- **Gender** — Patients' gender identity
- **ScheduledDay** — a day when an appointment is planned to occur.
- **AppointmentDay** — a real date of an appointment
- **Age** — a patient's age.
- **Neighborhood** — a neighborhood of each patient

- **Scholarship** — Does the patient receive a scholarship?
- **Hypertension** — Does the patient have hypertension?
- **Diabetes** — Does the patient have Diabetes?
- **Alcoholism** — Is the patient Alcoholic?
- **Handicap** — Patients' level of Handicap?
- **SMS_received** — Has the patient received an SMS reminder?
- **No_show** — Has the patient decided not to show up?

Source of the dataset -
https://www.kaggle.com/joniarroba/noshowappointments

Data Cleaning and Transformation

Data cleaning and transformation is a vital step as it makes data easier to investigate and build visualizations around. Below are few steps performed in this project to clean and transform data as required: -

- Using Pandas to read CSV file and use parse_date to convert data type of date columns from Object to datetime64

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/KaggleV2May2016.csv',parse_dates=['ScheduledDay','AppointmentDay'])
```

- Check sample of 10 rows data using head command

```
df.head(10)
```

- Dropping unnecessary fields as they are not required in the analysis

```
df.drop(['PatientId','AppointmentID'], axis=1,inplace=True)
```

- Check the structure of the dataset

```
df.info()
```

- Check number of samples and features of dataset using shape method

```
print("Number of patients: ", df.shape[0])
print("Number of data points(columns): ", df.shape[1])
```

- Rename misspelled columns – there are couple of typing errors in the column names. We use rename method to change the name of columns as shown below.

```
df = df.rename(columns={'Hipertension': 'Hypertension', 'Handcap': 'Handicap', 'SMS_received': 'SMSReceived'})
```

- Change No-Show column values for better understanding

```
df['Presence'] =df['Noshow'].apply(lambda x: 'Present'if x == "No" else 'Absent')
```

- Swap Appointment Day with Scheduled Day if difference is < -1. Sometimes the data can be inconsistent. If an appointment day comes before the scheduled day, then we need to swap their values.

```
df['AppointmentDay']= np.where((df['AppointmentDay']-df['ScheduledDay']).dt.days < 0, df['ScheduledDay'], df['AppointmentDay'])
```

- Calculate Wait Time and Weekday of appointment for each patient

```
df['Wait Time'] = df['AppointmentDay'] - df['ScheduledDay']
df['Wait Time'] =df['Wait Time'].dt.days

df['WeekDay'] = df['AppointmentDay'].apply(lambda x: x.weekday())
replace_map = {'WeekDay': {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday', 5: 'Saturday'}}
df.replace(replace_map, inplace=True)
```

- Check for Null values

```
df.isnull().sum()
```

Data Analysis and Visualization

In this step, I attempt to answer key questions listed below: -
Part 1 – Numeric Analysis

1. What is the gender wise distribution of patients who did not show up for the appointment?

```
df.query('Presence=="Absent"')['Gender'].value_counts()
```

```
F    14594
M     7725
Name: Gender, dtype: int64
```

2. What is the gender wise distribution of patients who were present for the appointment?

```
df.query('Presence=="Present"')['Gender'].value_counts()
```

```
F    57246
M    30962
Name: Gender, dtype: int64
```

3. What is the average age of the patient with respect to no-show data?

```
present = df['Presence'] == "Present"
absent = df['Presence'] == "Absent"

df_present_patients = df[present]
df_absent_patients = df[absent]

x1 = df_present_patients.Age.values
x2 = df_absent_patients.Age.values

print('Average age of people who showed up for the appointment :',round(average(x1)), 'Yrs')
print('Average age of people who did not showed up for the appointment :',round(average(x2)), 'Yrs')
```

```
Average age of people who showed up for the appointment : 38 Yrs
Average age of people who did not showed up for the appointment : 34 Yrs
```

4. Find the number of people who did not show-up for the appointment having multiple diseases and the age is above the average?

```
Q1= df.loc[(df['Presence_binary']==0) & (df['Hypertension']==1) & (df['Diabetes']==1) & (df['Age'] > average(df['Age']))
]

print('Number of people with NOSHOW having Diabetics, Hypertension and Age above average are:',len(Q1))
#Q1
```

Number of people with NO-SHOW having Diabetics, Hypertension and Age above average are: 1110

5. Find the number of people who did not show-up for the appointment as they did not receive SMS alert, wait time less than average wait time and the appointment day is on weekend.

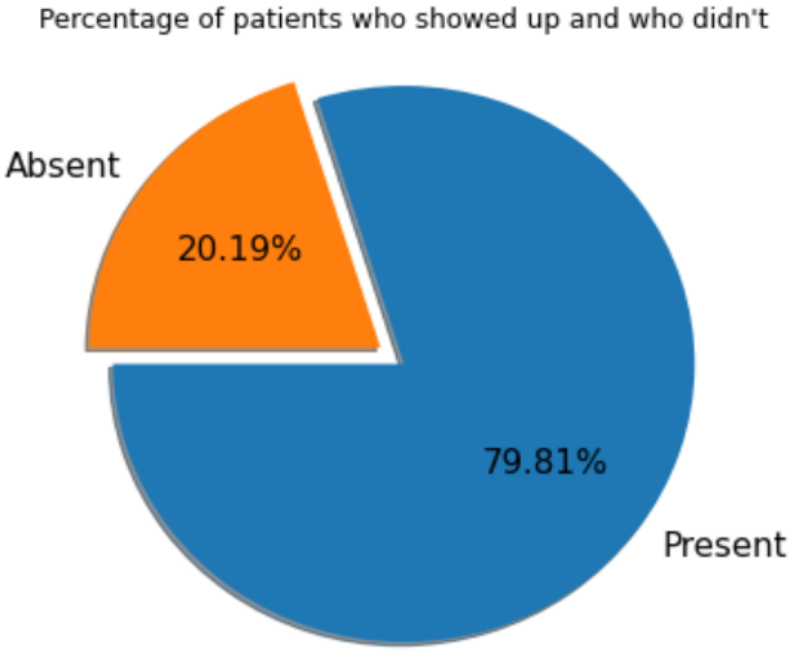
```
Q2 = df.loc[(df['Presence_binary']==0) &(df['Wait Time'] < average(df['Wait Time'])) & (df['SMSReceived'] == 0) & (df['WeekDay'] == 'Saturday') |  (df['WeekDay'] == 'Sunday')]

print('Number of people with NOSHOW having less than average Wait Time, did not receive appointment SMS alert and appointment day on weekend are: ', len(Q2))
#Q2
```

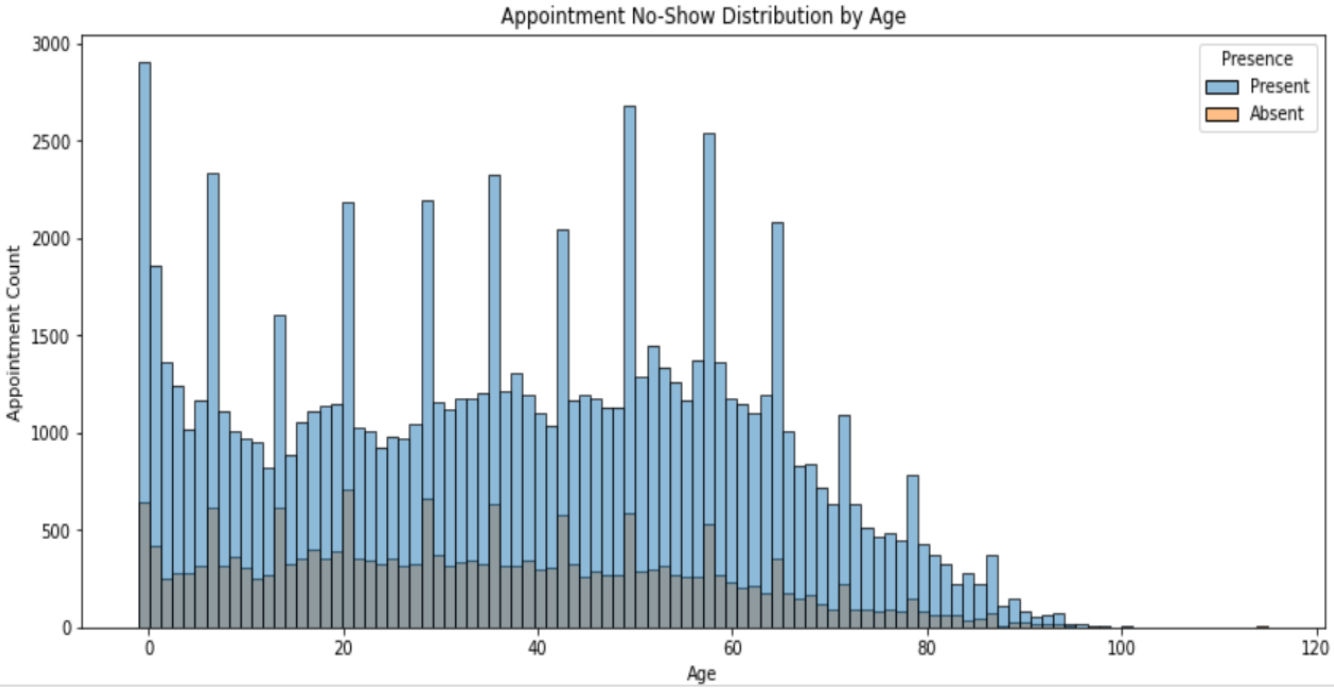
Number of people with NO-SHOW having less than average Wait Time, did not receive appointment SMS alert and appointment day on weekend are: 6

Part 2 – Data Visualization

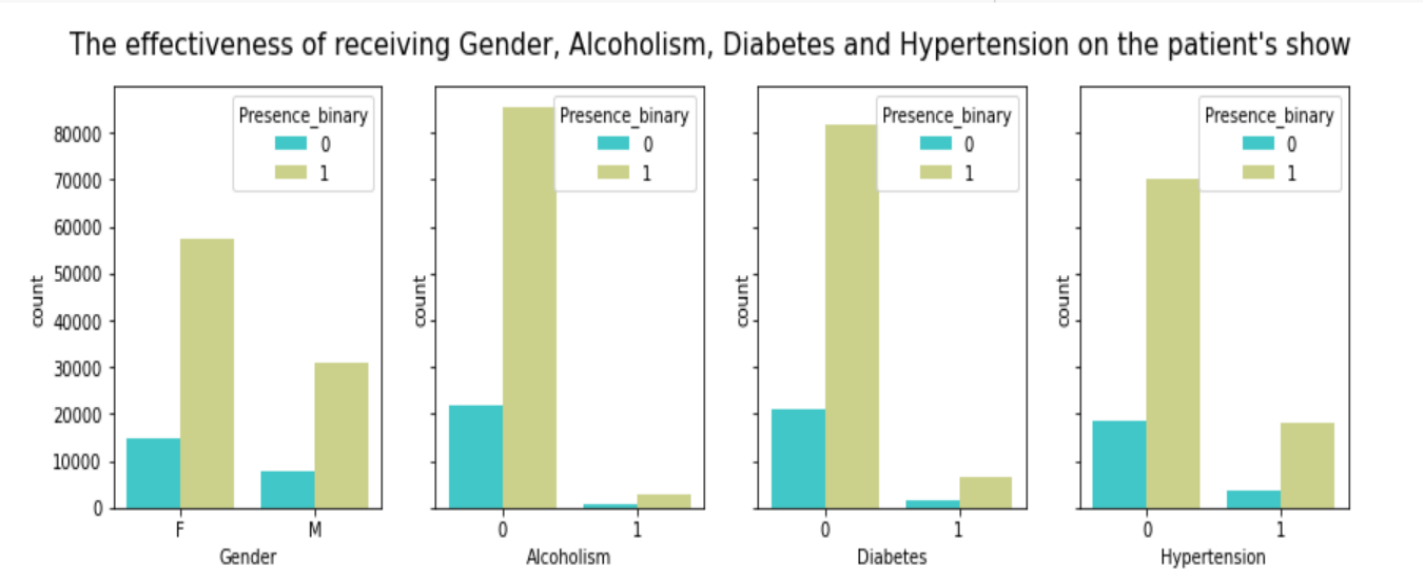
1. Show the graphical representation of patient percentage with respect to no-show data.



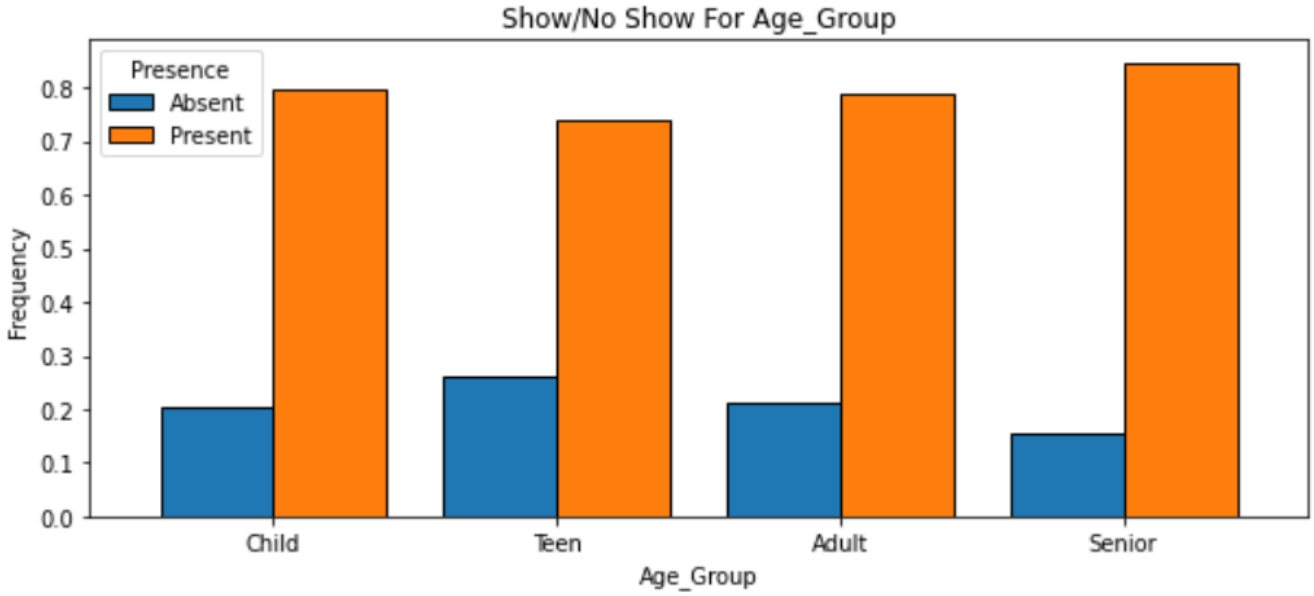
2. Visualize no-show data on top of appointment count and patient's age.



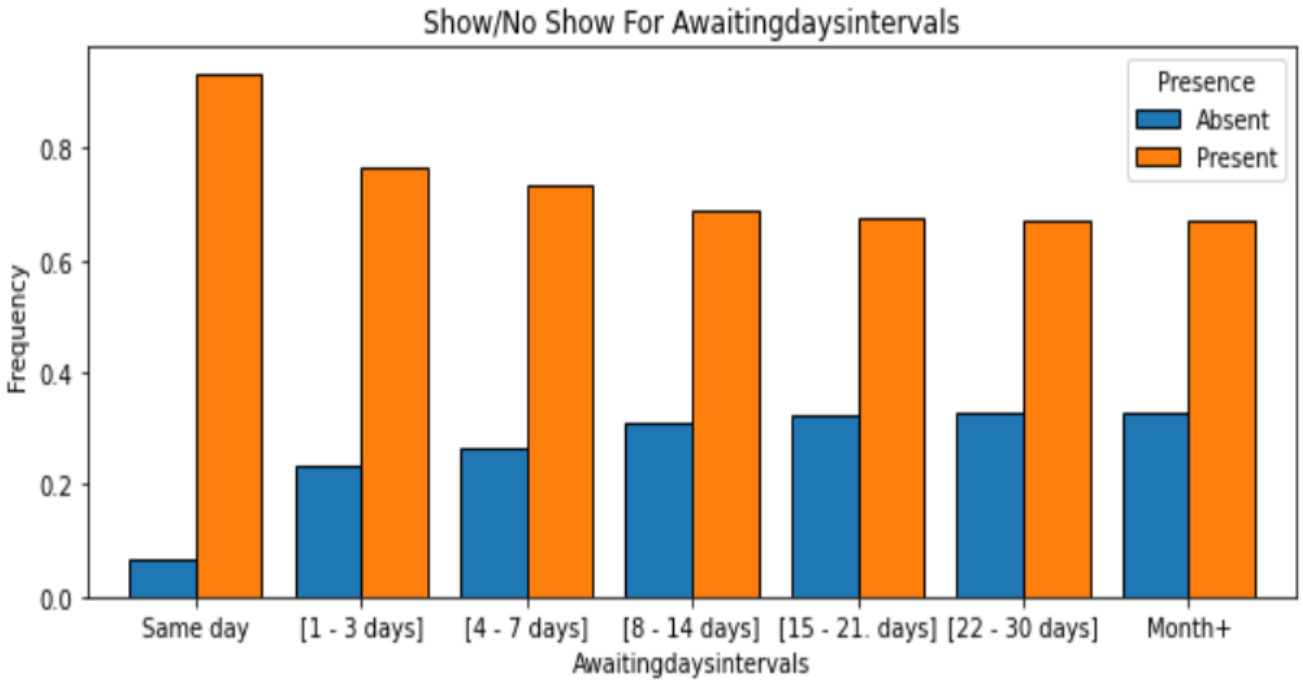
3. Show the effect of Gender, Alcoholism, Diabetes and Hypertension on No-Show data.



4. What is the probability of no-show with respect to patient's age?



5. Show the significance of Wait Time on no-show.



Program Description

Below attached is the Jupyter notebook consisting of all the executable commands in Python. I have added the text blocks in each command to better understand the dynamics of each code block.

mini_proj1_nivesh_final.ipynb

Result Analysis

1. Gender is not significant variable to determine the probability of Show/No Show. No-show percentage amongst patient is approx. 20% irrespective of the gender.
2. Age Group is a significant variable for the probability of Show/No Show. Teenagers are most likely to miss the appointment date, followed by children's and adult. Senior citizens are less likely to miss out on an appointment.
3. The probability of attending appointment decreases during Weekends. There are only 6 patients who missed an appointment on Saturday as they did not receive an SMS alert as well as the wait time was less than average time.
4. The Wait Time is a great significant variable to determine the probability of Show/No Show. As the wait time increases, the probability of missing out on an appointment increase.